Strategies for Evolving Diverse and Effective Behaviours in Pursuit Domains

Tyler Cowan

Department of Computer Science

Submitted in partial fulfilment
of the requirements for the degree of

Master of Science

Faculty of Mathematics and Science
Brock University, St. Catharines, Ontario

Abstract

Evolutionary algorithms have a tendency to overuse and exploit particular behaviours in their search for optimality, even across separate runs. The resulting set of monotonous solutions caused by this tendency is a problem in many applications. This research explores different strategies designed to encourage an interesting set of diverse behaviours while still maintaining an appreciable level of efficacy. Embodied agents are situated within an open plane and play against each other in various pursuit game scenarios. The pursuit games consist of a single predator agent and twenty prey agents, with the goal always requiring the predator to catch as many prey as possible before the time limit is reached. The predator's controller is evolved through genetic programming while the preys' controllers are hand-crafted. The fitness of a solution is first calculated in a traditional manner. Inspired by Lehman and Stanley's novelty search strategy, the fitness is then combined with the diversity of the solution to produce the final fitness score. The original fitness score is determined by the number of captured prey, and the diversity score is determined through the combination of four behaviour measurements. Among many promising results, a particular diversity-based evaluation strategy and weighting combination was found to provide solutions that exhibit an excellent balance between diversity and efficacy. The results were analyzed quantitatively and qualitatively, showing the emergence of diverse and effective behaviours.

Acknowledgements

Writing a thesis during a pandemic is no easy task. As the lines between work, school, and life became blurred, focusing on what needed to be done became increasingly difficult. Fortunately, I had the assistance of many wonderful people throughout my research. It's thanks to all of you that I was able to complete this thesis.

First, I'd like to thank my supervisor, Dr. Brian J. Ross, for recommending that I consider enrolling as a Master of Science. I am forever grateful for him taking me under his wing, teaching me what he knows about the concepts used throughout this research, and providing me with continual support and assistance.

I'd like to thank Illya Bakurov, Michael Gircys, and Doug Ord for their contributions toward the implementation of my systems. Their experience in genetic programming, multi-objectivization, and game-development saved me plenty of time during the development stages of this research.

I'd like to thank my friends, Mitchell Clark and Jordan Maslen. The three of us met within our first month as post-secondary students and we road out the entire journey together. I have absolutely no idea where I would be today had I not met you two, so, for the many all-nighters and crunch sessions, thank you.

I'd like to express heartfelt gratitude to my significant other, the most incredible woman I have ever met: Sara Testani. Your endless support and encouragement has helped me in so many ways throughout this thesis, and you've helped me find myself throughout this incredibly stressful pandemic. *Thank You.*

I'd like to extend a thank you to my family for supporting me and providing me with the resources and advice to get me to where I am. Thank you for putting up with my non-existent sleep schedule and my jargon-filled rants. You've helped me in more ways than I can write.

Lastly, I'd like to thank each and every one of the rest of you who have affected me in any way at all during my academic life. Every single moment, big or small, has led me to where I am today, and I couldn't be happier, so, thank you.

Tyler Cowan

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter introduces concepts about the search for interesting behaviour, intelligent agents, diversity search, and multi-objectivization, along with the objectives and motivation behind this research. Following that, the structure of this thesis is explained with brief descriptions of each chapter.

1.1 The Search for Interesting Behaviour

In artificial intelligence (AI), an evolutionary algorithm (EA) is typically designed with explicit goals like minimizing a price or maximizing a score. When the EA finds a particular combination of behaviours that achieve the desired goal, it is often the case that the evolved behaviours will promptly become exploited and overused. Due to the nature of fitness-based evaluation, given a sufficient period of evolution, the same set of evolved behaviours will typically be found by the EA. This is not a problem in most applications, as the means to achieve the goal are often unimportant, as long as the goal is achieved. In some applications such as video-games, however, succeeding in one static goal may be just one aspect of a desirable solution.

1.1.1 Intelligent Agents and Games

Intelligent agents are autonomous entities with the ability to sense and interact with their environment. Examples of domains that intelligent agents could be utilized in would include *herding* [1] [2], *food gathering* [3] [4], *predator versus prey* [5] [6] [7], and others [8] [9] [10] [11] [12] [13]. In a video-game application, an intelligent agent would be any non-playable character. Wilson *et. al.* were successful at evolving agents with simple but effective behaviours in a variety of 2D video-games [14]. Bullen and Katchabaw were successful at applying EAs to evolve effective agent behaviours in modern 3D video-games [15]. Bullen and Katchabaw noted that future work should place importance on the production of enjoyable and satisfying experiences for the players [15].

Intelligent agents and the search for interesting behaviour can be seen in Namco's classic arcade game *Ms. Pac-Man* [16]. Ms. Pac-Man tasks the player with consuming all of the food pellets within a maze-like arena whilst avoiding several predatory ghosts (intelligent agents). As per the rules of the game, the goal of any given ghost is to simply catch the player. The ghosts are hand-crafted to move randomly and only engage in beeline chases for short periods of time when the player is nearby. This behaviour is interesting and adds an entertaining level of unpredictability to the game whilst maintaining the challenge of avoiding the ghosts, but this comes with two significant caveats: (1) developers must spend time designing the behaviours of the ghosts, and (2) players will eventually learn and grow tired of the same behaviours being exhibited by the ghosts.

If a ghost were to be evolved using a traditional EA, a typical approach would involve rewarding the minimization of the distance between the ghost and the player. From this, the optimal play-style for the ghost would evolve to become a ruthless beeline chase toward the player at all times. The unfortunate consequence from this type of behaviour is an unfair and monotonous game. There exist many works that attempt to remedy this problem. Liberatore *et. al.* explore evolution occurring in real-time [17]. While their work showed promise, compromises had

to be made in order to ensure that the real-time evolution did not negatively impact the player's device too drastically. Dockhorn and Kruse explore evolution occurring for each ghost separately [18]. Their work was successful at adding diversity, but required multiple agents, where each agent would exhibit the issue of monotony on its own. Alhejali and Lucas explore the use of evolution occurring throughout multiple environments [19]. Agents could successfully adapt to the new environments, but would still act with monotony among the same environments. Rohlfshagen and Lucas introduce a competition to encourage improvements in general [20]. Competitions like this are great for generating interest in regards to the presented issue, but participants are scored on how successful their agents are, not how *fun* their agents are.

Unfortunately, this area of research still has a long way to go. As stated by Lehman and Stanley, *"evolving creatures in virtual worlds tend to converge to a single morphology because selection therein greedily rewards the morphology that is easiest to exploit"* [21]. If an agent is evolved with a traditional EA, it will most likely behave in a methodical, exploitative, and monotonous way, even between separate evolutionary runs [21]. This discourages developers from using "real AI" agents and limits them to what most refer to as "game AI" agents: simple, hand-crafted behaviour trees. These manually programmed opponents have the potential to avoid the pitfalls of a traditional EA, but will eventually succumb to their own pitfalls of monotony and predictability. Even if an EA is capable of producing an interesting or human-esque behaviour, like in Grossi's work [7], subsequent evolution runs often produce the same behaviour, and the player will eventually grow tired of it.

This leads to another issue in the search for interesting behaviours: how the agent behaviours make the player *feel*. As stated by Aslam and Brown, *"Emotions and perceptions are associated with each play experience. Therefore, players potential emotional outcomes must be regarded at an early stage of game design"* [22]. This should act as a point of guidance for those interested in researching the role of EA evolved opponents in video-games, since the quality of the opponents

has significant influence on the player's experience. As explained in a hypothetical situation by Aslam and Brown, "excitement", "curiosity", and "engagement" are examples of emotions that a developer would want their players to feel [22]. On the contrary, invoking feelings like "disappointment" and "helplessness" should be avoided [22]. In the case of an opponent possessing a diverse set of effective behaviours, one would expect the player to feel excited and engaged to be paired against such a variety of play-styles, and to feel curious in regards to what other play-styles might be revealed later on. In the case of an opponent possessing a monotonous set of behaviours, one would expect the player to feel either disappointment (if the behaviours can be learned and conquered by the player) or helplessness (if the behaviours are too difficult to be learned and conquered by the player).

Ultimately, the search for interesting behaviours needs more work. From what has been explained thus far, in order to find interesting behaviours, it only seems natural to also seek *diverse* behaviours. No matter how interesting a particular behaviour may first appear, continued exposure will inevitably dull the interest. As such, it can be stated that the search for interesting behaviour relies on the search for novel behaviour.

1.1.2 Diversity Search

The search for diversity is a relatively young concept, but it has been gaining traction over the past decade in particular. Some early approaches to the concept are explored by Lehman and Stanley [21] [23] [24] and Mouret and Doncieux [25] who have produced works in which diverse behaviours outperform their fitness-focused counterparts. With this barrier broken, diversity search has grown, and will hopefully bring evolutionary computation down a new path of simulated creativity. The desire for diverse behaviours has already reached commercial desire in video-game applications [26].

Works from Lehman and Stanley [21] [23] [24] and Mouret and Doncieux [25] explore *novelty search*, and later papers begin exploring additional approaches to

achieving diversity. These techniques work to solve the deceptive maze problem; agents must navigate through a maze and reach the end. The problem is inherently difficult for fitness-based evaluation techniques, as maze solving typically requires the agent to move further from the obvious end-goal (distance to the exit) to explore the maze thoroughly. Fitness-based evaluation techniques rely on measurements like the distance between the agent and the end-goal, so the nature of fitness-based evaluation quickly rules out the thorough exploration in favour of getting stuck in corners. The diversity-based evaluation strategies prove that one can get around this problem by encouraging diverse behaviours among the population.

These diversity strategies seem promising but the domains in which they are explored leave little room for interesting behaviours to emerge. In order to properly dive into the search for interesting behaviour, a domain with room for creativity needs to be used.

Additionally, Mouret [27] explores multi-objectivization and its contribution to effective diversity. With the use of multi-objectivization, multiple behaviours can be measured to encourage diversity, as well as fitness-focused behaviours. While Mouret used the popular *Pareto ranking* [28], this research will opt to use *sum of ranks*; details on that decision can be found in Chapter 4.

An important note to make when considering the search for diversity is the shift of the overall objective. Fitness-based evaluation relies on a static objective, while diversity-based evaluation is more open-ended [29]. This open-endedness results in a much larger search space that might contain vast regions of undesirable behaviours. Gomes *et. al.* explains that this problem is *"overcome by combining exploratory pressure of novelty search with the exploitative value of fitness-based evolution"* [30]. In some works [31], it has been found that diversity *alone* can achieve effective results, sometimes even more effective than the fitness-based evaluation counterparts [31].

1.2 Objectives and Motivation

In the world of embodied agents and evolutionary algorithms, the level of interest provided by an emergent behaviour is often overlooked. This leaves the more creative-focused domains lacking in progress. From what was mentioned in Section 1.1.1, the predator versus prey domain is a great place to explore the balance between interesting and effective behaviours, and from what was mentioned in Section 1.1.2, diversity and multi-objectivization would be helpful, if not necessary, to include. Considering the subjectivity of what one might consider "interesting", the term will be regarded in the following way: if a behaviour is more *novel* (diverse), then it is more interesting.

A large portion of games involve predator versus prey situations. An example providing significant inspiration for this research is the popular video-game *Alien: Isolation* [32]. This research could potentially accelerate the creation of similar games by reducing the labour time involved with developing interesting opponents. Additionally, other portions of game development (play-testing, for example) could become more automated through the use of diverse agents.

Games are not the only domain that may benefit from this research, however. Diversified behaviours have the potential to improve a wide range of applications, like art, architecture, virtual robotics, and so on [33]. Even in domains like these, a thoroughly evolved population of solutions will tend to provide results that are the same, or at least recognizably similar. Hopefully, this research can help remedy that. This leads to the goals of this research:

1. **Combine Fitness and Diversity in Pursuit Domains**: The first objective of this research is to combine the traditional fitness-based evaluation of a solution with a diversity-based evaluation strategy. Four diversity strategies are explored throughout this research, each of which being inspired by Lehman and Stanley's [29] novelty search. Diversity will be calculated using the sum of ranks of four behaviour measurements. The resulting value will then be combined with the original fitness value to produce the final fitness

for the solution.

2. **Produce Diverse Behaviours with Minimal Harm to Fitness**: The second objective is to ensure that fitness goals are not sacrificed too drastically. As stated by Bullen and Katchabaw, *"In the end, it is important to also determine if the evolved [agents] deliver a more enjoyable and satisfying experience to a human player"* [15]. Producing solutions using traditional fitness as the sole evaluation strategy should lead to sets of monotonous and uninteresting behaviours, while producing solutions using diversity as the sole evaluation strategy should lead to ineffective and nonsensical behaviours. In order to ensure the agents can provide an enjoyable and satisfying experience, a balance must be found to produce sets of solutions that are noticeably diverse with minimal negative impact to the original fitness score.

3. **Observe the Behaviours Quantitatively and Qualitatively**: The final objective is to observe the resulting behaviours quantitatively and qualitatively. This includes programmatically measuring the diversity among solutions, watching the real-time replays of solutions and recording observed behaviours, and analyzing the resulting data for statistical significance.

These can be rephrased into the primary objective of this thesis: to evolve embodied agents in such a way that diverse and interesting behaviours can emerge whilst maintaining an appreciable level of efficacy, specifically in a domain that allows for creative solutions. This will be achieved by using a genetic program (GP) as a predator agent controller, evolved with a combination of multi-objectivized diversity (novelty search) and fitness. In other words, this thesis will explore *strategies for evolving diverse and effective behaviours in pursuit domains.*

1.3 Thesis Structure

Following this introduction (Chapter 1), the thesis will dive into the background reading (Chapter 2) involved in the foundation of this research. This will talk in

detail about the major concepts utilized throughout this research.

Following that will be two chapters on design decisions (Chapters 3 and 4), in which the implementations of the previously discussed concepts are explained, along with other design decisions. Specifically, Chapter 3 will describe the systems used throughout this research and why they were developed the way they were, while Chapter 4 will describe details common to every experiment.

Next will be the primary importance of this research: three experiment series chapters (Chapters 5, 6, and 7). Each of these chapters will show a fitness-based baseline experiment, several diversity-based comparison experiments, and a discussion for analysis. A concluding chapter (Chapter 8) summarizes the research and discusses omitted and future work.

For layman readers, Chapter 3 can be skipped, while Chapter 2 will be of particular importance. For readers experienced in these topics, Chapter 2 can be skipped, while Chapter 3 might be of interest. In either case, chapters 5, 6, and 7 are the focus.

Chapter 2

Background Reading

This chapter describes the ideas and concepts that drove this research. The concepts are explained in detail but their use in this research is described in Chapter 3. This chapter should give a clear understanding of the foundations of this research.

2.1 Genetic Programming

A genetic program (GP) is an evolutionary algorithm (EA) with which a population of programs is evolved to solve a particular problem [34][35]. The population typically initializes with randomized programs (individuals) and then follows a nature-inspired evolutionary process of breeding and mutating throughout many generations. After enough generations, the population will hopefully converge to an optimal or near-optimal behaviour. The best individual is then taken from the last generation and used as the solution to the problem.

There exist many domains in which GPs can be used to solve a problem, such as medical classification, image recognition, symbolic regression, and in the case of this research, embodied agent controlling.

2.1.1 Inside a Genetic Program

A GP individual is structured as a tree, where each non-leaf node is an operator (a function; one or more arity) and each leaf node is an operand (a terminal; zero arity). A literal tree representation is the human-friendly way to interpret an individual, but the GP system will obviously handle the individual in a code format (LISP, for instance). As an example, the GP might be tasked to produce a math function like the following:

$$f(x, y) = 27 - \frac{y}{0.5} + \cos x$$

A possible tree representation of a GP's solution is shown in Figure 2.1. In this example, the nodes marked "27", "Y", "0.5", and "X" are terminals; they have zero arity (take no arguments). The rest of the nodes are functions that require arguments, and therefore have some arity above zero. It is common for the terminals to be the input for the GP and for the root to be the output. In this example, we have two variable inputs (X and Y) and two constant inputs (27 and 0.5). After the tree has finished executing, the root will hold the solution. An example of a code representation of the GP's solution is shown in Figure 2.2.

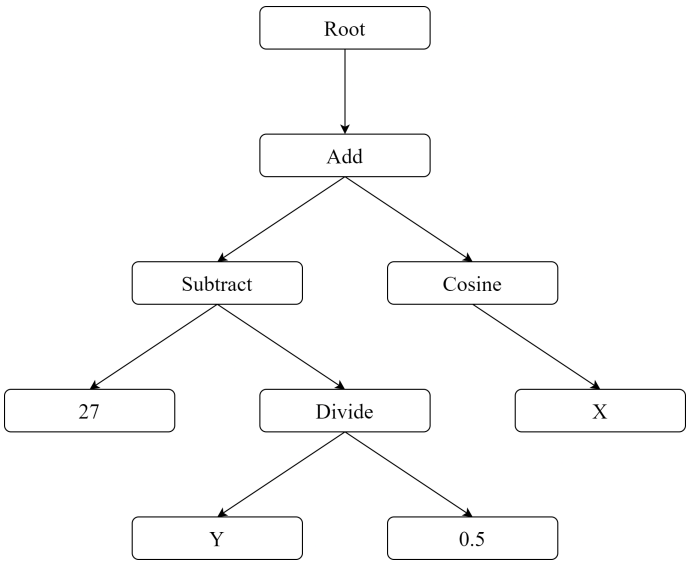


Figure 2.1: Simple Generic Genetic Program - Tree Representation

```
Root (
    Add (Subtract (27 Divide(Y 0.5)) Cosine(X))
)
```

Figure 2.2: Simple Generic Genetic Program - Code Representation

This represents the exact same program as Figure 2.1, but in a computer-friendly manner. This is what is used in the back-end of the GP system (and is the format used throughout this research). Actual programs evolved during this research can be seen in the appendices, represented in their code format as well (the tree representation would be far too large).

Another example more related to this research can be seen in Figures 2.3 and 2.4. In this example, the individual is comprised of six nodes; three functions (Root, Subtract, and Desired Location) and three terminals (Pred Location, Desired X, Desired Y).

In more detail, the "Desired X" and "Desired Y" terminals would return one-dimensional values to the "Desired Location" function, which would combine them into a two-dimensional vector. The "Subtract" function would then subtract the vector returned by the "Pred Location" terminal from the desired location vector.

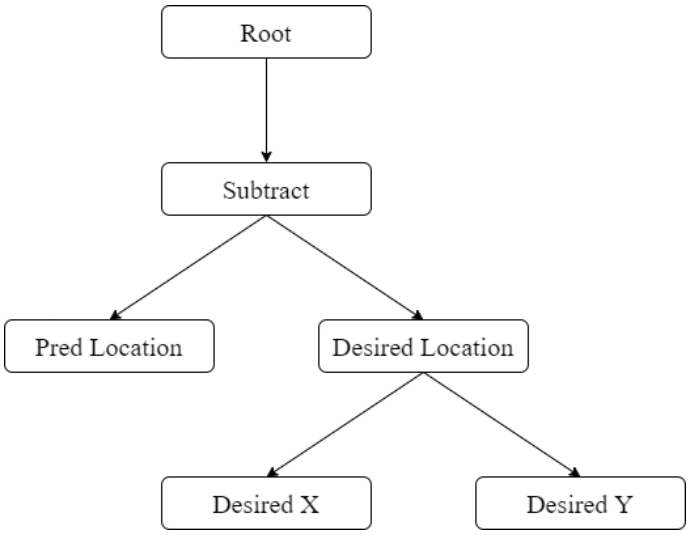


Figure 2.3: Simple Predator Genetic Program - Tree Representation

```
Root (
    Subtract (
        PredLocation(), DesiredLocation (
            DesiredX(), DesiredY()
        )
    )
)
```

Figure 2.4: Simple Predator Genetic Program - Code Representation

The result would be the two-dimensional vector from the predator's location to the desired location, or in other words, the look-at rotation vector. This would then be passed to the root to be used as the new rotation for the predator it controls.

2.1.2 Evolving a Genetic Program

To achieve programs like the ones previously shown, the population of individuals are put through the evolutionary process and modified by genetic operations (inspired by natural evolution) over many generations. Some key components of this process are as follows:

1. **Fitness**: Each individual in the population is given a "fitness" value. The fitness is used to determine how effective the individual's ability at solving the provided problem is. In order to calculate the fitness, some evaluation strategy needs to be designed specifically for the problem at hand.

2. **Selection**: During portions of the evolutionary process (crossover, for example), individuals will need to be chosen based on their fitness, in which more fit individuals are more likely to be selected. The selection is often done through a tournament in which a set of k random individuals are compared. The individual with the best fitness value is typically deemed the winner of the tournament and will be selected.

3. **Crossover**: Pairs of individuals are used as parents to breed new individuals. This operation typically provides two new individuals for the next generation,

each with a mix of genetic code from their two parents (typically done by swapping randomly selected sub-trees between parents). As an example, two child individuals being prepared for a new generation might be initialized as copies of their parents, then both children will swap sub-trees of their genetic code. This permits inheritance of parent traits.

4. **Mutation**: This has two distinct uses in this research. The first usage involves selecting a portion of an individual's genetic code and replacing it with a newly generated portion. The second usage involves selecting a portion of an individual's genetic code and slightly modifying the terminal values within it.

5. **Elitism**: When a new generation's population is being created, it might be desired to keep the previous generation's best solution(s). Elitism will ensure that unmodified copies of the best individual(s) will be brought into the next generation. Typically, only a few elite individuals are saved during each generation.

Most GPs will utilize most, if not all, of the above mentioned components. Likewise, the evolutionary process in which these components are employed will typically flow in the same manner as outlined in Figure 2.5's pseudo-code.

The focus of this research revolves around a specific part of this evolution

```
Begin
Generate initial population of individuals
For each generation, or until convergence:
    Compute fitness of all new individuals
    Select individuals from old population
    Crossover individuals from selection to create new individuals
    Mutate new individuals
    Bring previous population's elites into new population
Return best individual
End
```

Figure 2.5: GP Pseudo-code

process: computing the fitness. As previously mentioned, in order to calculate the fitness for an individual, some evaluation strategy needs to be used.

2.2 Evaluation Strategies

In order to determine how effective an individual is at solving the presented problem, the fitness needs to be calculated. To calculate the fitness, it is required that an evaluation strategy be crafted with respect to the target problem.

This research will refer to two types of evaluation: traditional "fitness-based" evaluation, and the newer "diversity-based" evaluation. In either case, the evaluation will eventually provide a single value that will be used as the fitness.

2.2.1 Fitness-based Evaluation

The traditional technique for evaluating an individual is with fitness-based evaluation. Fitness-based evaluation utilizes a performance metric defined by the user to compare individuals and their efficacy in solving the problem at hand. In some applications, the value must be minimized, while others require it to be maximized.

An example of a minimization problem, the maze problem, tasks the GP with navigating and escaping a maze. The closer the individual gets to the exit, the smaller the distance value would be. The fitness-based evaluation for this problem could simply be measuring the distance between the individual and the exit from the maze. Individuals with smaller distances would be valued higher than individuals with larger distances.

An example of a maximization problem, predator-versus-prey, tasks the GP with catching as many prey as possible. The more prey that an individual catches, the better the score will be, so the fitness-based evaluation could simply be the number of prey caught by an individual. Individuals who catch more prey would be more valuable since their score is higher.

However, fitness-based evaluation is not a perfect system. Premature convergence is a common problem among fitness-based evaluation systems, ultimately

leading to sub-par solutions if not dealt with appropriately. Premature convergence occurs when the population gets "stuck" on a particular behaviour. This is a significant drawback when solving tasks like the maze problem, as discussed in Chapter 1.

2.2.2 Diversity-based Evaluation

Diversity-based evaluation is a relatively young technique for evaluating individuals [25][29][23]. The idea behind it is this: instead of evaluating an individual based on some objective measurement, the solution is evaluated based on how *unique* it is compared to other solutions. It has been shown that searching specifically for diverse solutions can lead to results that out-perform those found through fitness-based evaluation [23].

An implementation of diversity-based evaluation, novelty search, is described by Lehman and Stanley [23] as an algorithm that *"searches with no objective other than continually finding novel behaviors in the search space"* [23]. This is achieved by comparing behaviours of an individual to an archive of past individuals who possessed highly novel behaviours at their birth. This is found to mitigate the problem of premature convergence, as any new behaviours with the potential to cause premature convergence will be added to the archive and avoided in the following generations. This has led to solutions to deceptive problems like the maze problem.

Yannakakis and Liapis [36] make a different approach to diversity-based evaluation: *surprise search*. Instead of trying to avoid previous solutions, like novelty search does, surprise search avoids *future* solutions. By using prediction models, Surprise Search extrapolates what a future solution may look like, and then reward an individual for avoiding it.

In either case, to determine the diversity of an individual, characterization vectors need to be calculated and compared. A characterization vector will be some n-dimensional vector comprised of n-many quantitative behaviour measures. For example, in Figure 2.6, two quantitative behaviours are used to form a two-

dimensional search space. In this example, each individual's characterization vector is plotted on a Cartesian plane, with each axis representing one of the two quantitative behaviour measures. The characterization vectors are assigned a colour based on how diverse they are compared to the others' behaviour measurements. The black vector is located at the average of all vectors, representing the least diverse characterization. As the vectors stray further from the average, their colour shifts from red to green, with green being the most diverse.

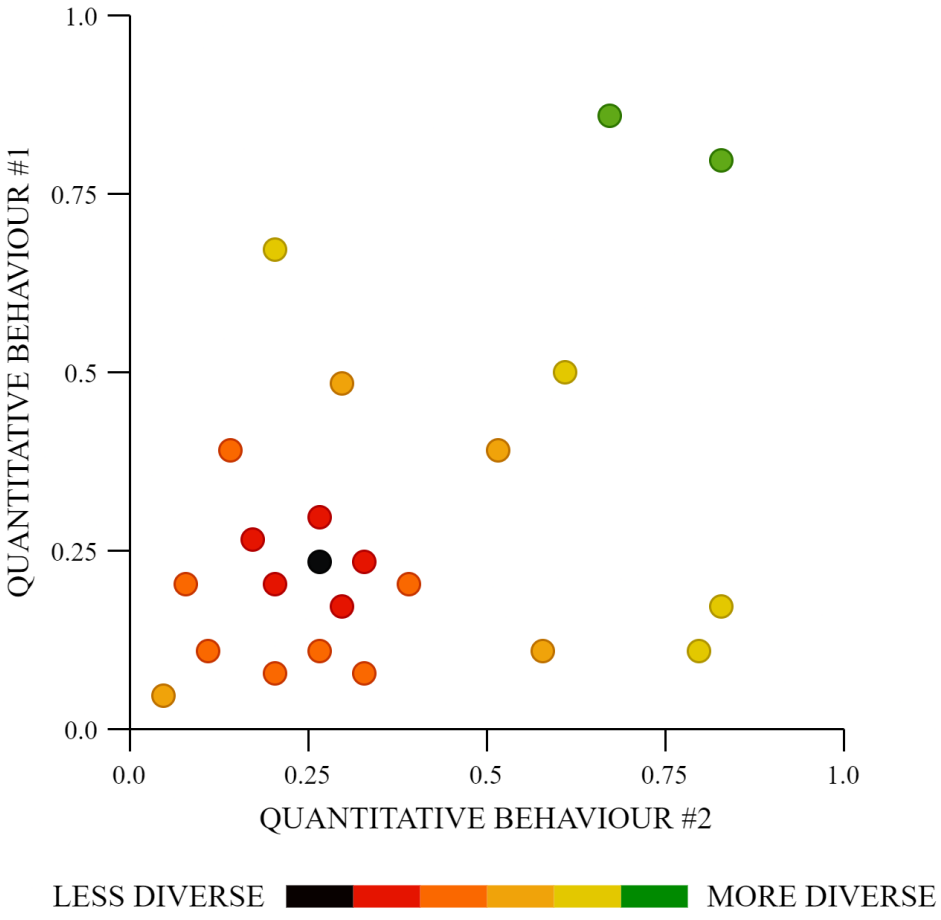


Figure 2.6: Example of Quantitative Behaviour Plot
Each data point represents a 2-dimensional characterization vector. Those closer to the average are "less diverse".

The assigning of colours is done simply by calculating the Euclidean distance between a characterization vector and the average. As such, the distance between the average and a red vector is small, while the distance between the average and a green vector is large.

2.3 Embodied Agents

Embodied agents are a subset of intelligent agents, which are autonomous entities with the ability to sense and interact with their environment. Examples of intelligent agents include AI home assistants and thermostats. A definition for agents offered by Panait and Luke [10] is:

> *"An agent is a computational mechanism that exhibits a high degree of autonomy, performing actions in its environment based on information (sensors, feedback) received from the environment."*

This definition, for all intents and purposes, is how an agent will be considered throughout this paper. Furthermore, the definition offered by Panait and Luke is expanded to describe a *multi-agent* environment as one where agents may interact with one another, and where information is limited such that agents may not know everything about the environment that other agents know [10].

Related works such as Gomes' competitive [37] and cooperative [38] evolution papers have a strong adherence to such definitions, but this paper focuses on the evolution of a single predator agent, so the prey agents only loosely follow such definitions. The structure of a simple intelligent agent can be see in Figure 2.7.

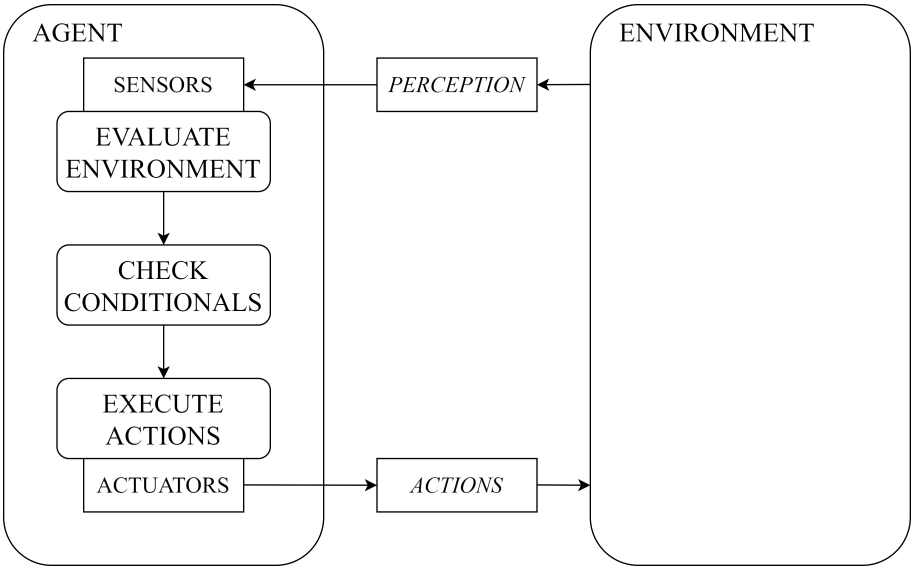


Figure 2.7: Intelligent Agent Diagram

While intelligent agents do not require a body, embodied agents do (as the name suggests). In a physical environment, an embodied agent could be a robot. In a virtual environment, an embodied agent could be any entity representable in a graphical way. Embodied agents will typically operate with the traditional deterministic automata paradigm. The agent's actions depend solely on what it currently senses, with no recollection of previous information gathered. The agent's body could occupy a space and have an effect on the environment it perceives, or in other words, the environment might change based on the behaviours exhibited by the agent.

2.3.1 Emergent Behaviour in Agents

Intelligent agents and emergent behaviour have been widely discussed concepts for many years now. A fascinating research project from 1994 shows embodied agents evolved through Darwinian evolution and the many emergent behaviours that resulted from it [39].

An important concept stemming from intelligent agents, *emergent behaviour*, concerns the evolution of advanced behaviours. As shown in Grossi's [7] work, a collection of primitive functions can eventually be combined to produce *"interesting behaviours that show high-levels of coordination among agents"* [7]. In Grossi's [7] work, the GP language was comprised of simple functions: cardinal direction movement, fixed rotations, and basic arithmetic are some examples. As the GP evolved the agents over many generations, interesting high-level behaviours would eventually emerge.

This is emergent behaviour, and it is one of the most important components required to successfully evolve interesting behaviours.

2.4 Multi-objectivization

Multi-objectivization represents the combining of multiple single-objective measures into one. These behaviours can be related in complex and often non-linear

ways. With one objective, the ranking of a solution is proportional to the objective, as in, if a solution is good at X, it would have a better rank than a solution that is bad at X. Multi-objectivization addresses the problem introduced by the existence of more than one objective.

2.4.1 Weighted Sum: Weighted sum is one of the simplest approaches to implementing multi-objectivization. With weighted sum, each objective is applied a weight to represent how valuable that objective is, then each weighted objective is summed together to form a single value: the score.

This method comes with a fault, however, as poorly decided weights may exaggerate or bury particular objectives. The following formula calculates the weighted sum of n-many objectives:

$$f(x) = w_1 o_1 + w_2 o_2 + ... + w_n o_n$$

Where w is the weight and o is the objective score, this formula will combine the value of every objective, scaled by the chosen weight, into a single score.

2.4.2 Sum of Ranks: Sum of ranks is a longer process but eliminates the need for a user to decide on weight values [40] [41]. Each solution is given a rank (some positive integer, 1^{st}, 2^{nd}, 3^{rd}, etc.) for each objective based on how well the score for the objective is compared to the rest of the solutions. After every individual has a rank applied for each of their objectives, the rank values are added together to make the final score for the solution.

For example, if an individual placed 1^{st} in one objective and 4^{th} in another, then their sum of ranks score would be 5. Again, this solution is not perfect. A solution could dominate in every category but severely under-perform in, what could be, the most important category.

An example of the process can be seen in Table 2.1. Row 1

2.4.3 Pareto Ranking: Pareto ranking is similar to sum of ranks in that it applies ranks for each objective relative to how well the solution performed against

Table 2.1: Example of Sum of Ranks Calculation [42]

	OBJECTIVES			RANKS						
ID	O1	O2	O3	o1	o2	o3	SUM OF RANKS	RERANKED	NORMALIZED	RERANKED
1	2.75	8.5	37.5	3	1	1	**5**	1	1.1333	**1**
2	1.75	0.5	12.5	4	5	3	**12**	3	2.8	**4**
3	0.25	3.5	12.5	5	4	3	**12**	3	2.8	**4**
4	4.25	5.5	12.5	2	3	3	**8**	2	2	**3**
5	12.25	6.5	22.5	1	2	2	**5**	1	1.2667	**2**

other solutions. The difference with Pareto ranking is that the ranks are not combined. Instead, each solution is given a final ranking based on how many of its objectives are "non-dominated" by other solutions [28].

For example, a solution with the best score in X and Y would be non-dominated and receive the highest rank. A solution with the best score in X and second best score in Y would receive the second highest rank. The drawback is when some other solution earns the best score in Y and second best score in X; that would also receive the second highest rank. In this case, two solutions are tied for second place, and had the first solution not dominate both categories, it would be up to the user to decide which of the other two is best.

This is known as the "Pareto frontier", and in some applications, it may be desirable to be provided with multiple solutions at the end of an experiment, but often times it is not. Another significant drawback is the ineffectiveness when dealing many objectives, as the extra dimensions lead to a set of solutions growing to an overwhelming size (unlike sum of ranks).

Chapter 3

System Design

This chapter discusses the system and its implementation for this research. It will describe in detail how specific features of the system functions and why they were chosen. This chapter should provide a clear understanding of how and why the system was created the way it was.

3.1 Continuous Environment with Discrete Overlay

Considering the nature of a pursuit-game scenario, it seemed fitting to provide the agents with a continuous environment. A continuous environment is an environment in which floating point values are used for the coordinates and speeds, rather than integer values; this applies to rotation too. This means the agents that move within it are free to move in any direction by any amount (unless restricted by other means, of course). Many papers limit their agents' movement to the four cardinal directions on a grid (discrete environment), but such a decision places a much more narrow boundary on the behaviour search space than a continuous environment would. On account of this research focusing on diversified behaviours, a continuous environment was the best option, freeing the agents to move in any direction.

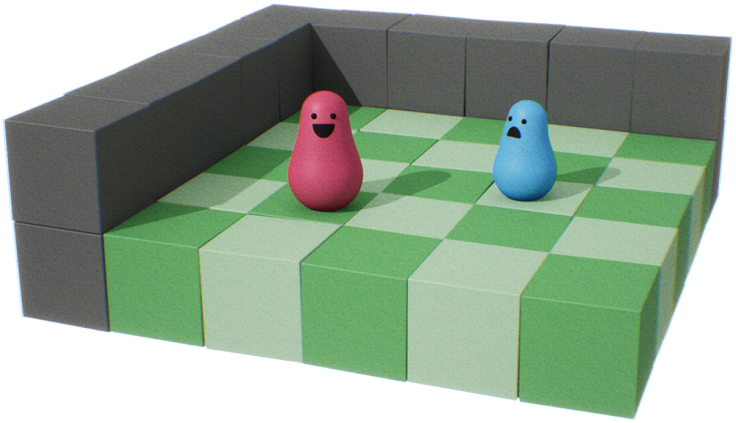


Figure 3.1: Environment with Agents
A 3D representation of a portion of the environment used throughout this research.

For the sake of simplicity, a discrete overlay is still used in tandem with the continuous environment. The environment is segmented into a grid to create the discrete overlay, where each cell is one from the following described in Table 3.1. The 3D representation of this environment can be seen in Figure 3.1. While the environment is visually segmented into a grid, (to represent the cell locations of the discrete overlay) the agents are free to move in any direction by any amount, as previously mentioned.

Each cell is one unit by one unit, and the environment has 200 units for length and 200 units for width. This brings the discrete overlay to a total to 40,000 distinct cells. By using a discrete overlay, information about the environment can be fed into the GP with relative ease. For example, if any traces need to be done throughout the environment, they can be simplified to checking the cells that the trace would overlap. This makes collision handling much more straightforward, and also simplifies the spatial awareness for agents within the environment.

Table 3.1: Discrete Overlay Cell Descriptions

CELL	DESCRIPTION
OPEN	An open area with nothing inside.
WALL	An obstacle that nothing may pass through.
PRED	The predator agent.
PREY	A prey agent.
HUSK	A previously eaten prey agent.

The furthest an agent can move during their turn is one unit. Therefore, the agent can never skip over a cell. An agent does not have to move a full unit, or it might be moving at some angle that would result in it landing in the same cell, despite moving a full unit.

As an agent navigates the environment, the cell it resides in is labelled appropriately: PRED for the predator and PREY for the prey. If any agent attempts to move into a cell labelled WALL, the move will be rejected and their turn is ended. Cells labelled OPEN or HUSK have no effect on the simulation: OPEN cells are simply empty, while HUSK cells are for book-keeping and visualization after the simulation is complete.

Every simulation runs for 5000 time units (ticks). During a tick, each agent takes its turn, starting with the predator. Once all the turns are completed, the next simulation tick begins. The predator's turn requires a single execution of the GP-evolved program. The resulting value is the predator's new rotation.

Simulations can be logged to produce a "replay" file. These replay files allow the simulations to be watched in a human-friendly fashion. A replay can be watched directly within the system or exported into a 3D game engine for a more

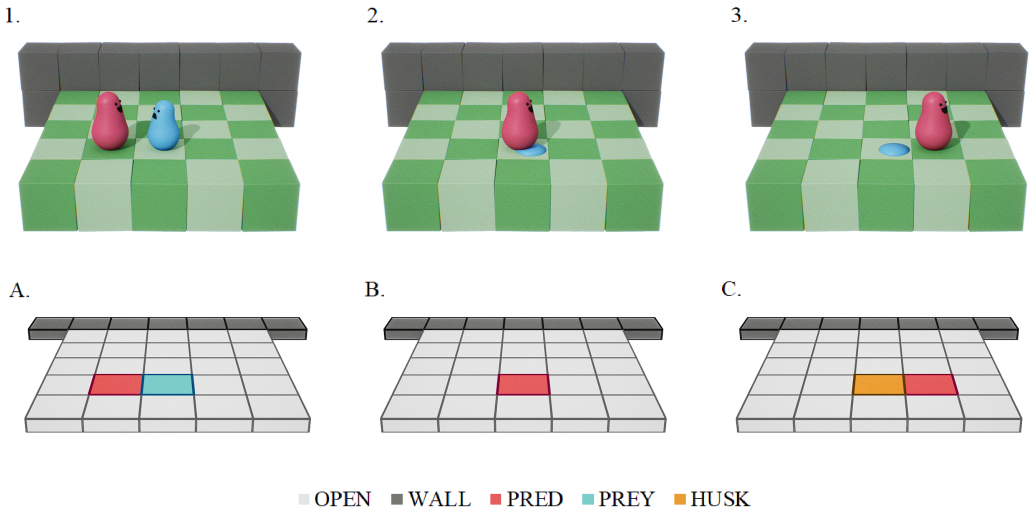


Figure 3.2: Continuous Environment with Discrete Overlay
Images 1, 2, and 3 show a predator catching a prey and moving on.
Images A, B, and C show the discrete overlay representation.
Legend corresponds to A, B, and C.

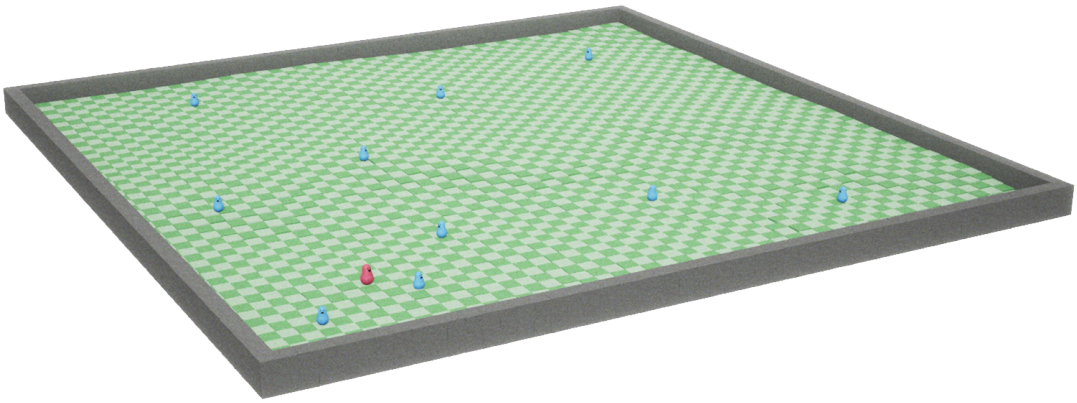


Figure 3.3: Simulation Environment
A 3D representation of the simulation environment, reduced for better viewing. The arena in this image is only 50 units long and 50 units wide (2500 total cells).

aesthetically pleasing experience. Figure 3.2 shows an example of the 3D representation of a replay, along with an illustration of the corresponding discrete overlay. The purpose of a replay is to allow for the collection of empirically identified behaviours (qualitative behaviours), and of course, because it is entertaining. Note that the illustrations use only five units for the length and five units for the width; this is only a small fraction of the actual size of the environment. A visualization of an arena (reduced to fit as a figure) used can be seen in Figure 3.3.

3.2 Predator and Prey Controllers

Each agent has a two-dimensional location vector (LOC) representing their coordinates within the environment, a two-dimensional rotation vector (ROT) representing the direction they are facing, and a speed value (SPEED) representing the distance they can move in a single tick. During the predator's turn, the predator will collect information on the environment and then feed it into its GP-evolved tree. The value returned by the tree will replace their ROT. The predator then rotates to match the direction of ROT and moves a distance in that direction equal to SPEED.

The prey behave in a similar manner, replacing the GP-evolved tree with their

own semi-constant rotation speed value (ROTSPEED). During each tick, the prey will rotate their ROT by ROTSPEED, and every five ticks, the prey will modify ROTSPEED by a small random amount. The result is an agent that appears to move in a natural but random motion which can be likened to the movement of a bee searching for a flower.

The predator has a speed of 1.0 unit per tick, while the prey have a speed of 0.7 units per tick. The predator has a field of few of 90 degrees with a distance of 20.0 units, along with a sensing radius of 5.0 units. If a prey enters either of these zones, the predator will become aware of its location for as long as the prey remains within them. If a prey enters and leaves the predator's field of view, the predator will remain aware of the prey's location for another five ticks. If a prey leaves the predator's sensing radius without ever being in the field of view, the predator will lose awareness of the prey immediately. A diagram of the predator can be seen in Figure 3.4.

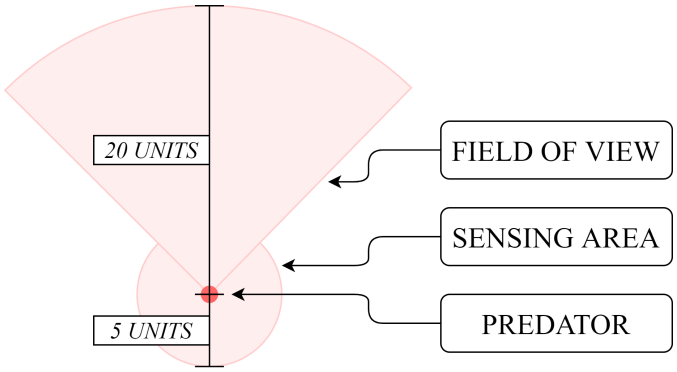


Figure 3.4: Predator Agent Perception
A diagram displaying the sight and sensing areas of a predator.

3.2.1 Predator Versus Prey

In this implementation of Predator Versus Prey, the predator must catch as many prey as possible before the simulation ends. If the predator takes a step and finds that it has landed within a cell labelled PREY, it will consume the prey within that cell and relabel it PRED. Similarly, if the prey takes a step and finds that

it has landed within a cell labelled PRED, it will be consumed by the predator within that cell. The same happens if more than one prey inhabit the same cell.

Each prey caught by the predator rewards the predator with one point and leaves behind a HUSK, regardless of whether the predator moved onto the prey or the prey moved onto the predator. The predator's fitness-score is determined by the number of prey it can catch before the 5000 ticks have passed. If all prey are caught, the simulation ends early.

3.2.2 Predator Versus Advanced Prey

Predator Versus Advanced Prey is the same as Predator Versus Prey, with a self-explanatory change: the prey are more advanced. An unfortunate drawback of the initial Predator Versus Prey was the lack of intelligence on the prey's behalf. Since they simply moved with an occasional modification to their rotation speed, they would often get stuck on walls for brief periods of time, or even move right into the predator. In this implementation, the prey are given two significant upgrades:

1. **Bouncing**: If a prey is adjacent to a wall and the next step would cause it to impact with the wall, it will immediately randomize its rotation. This effectively causes prey to "bounce" off of the walls. This simple upgrade almost completely removes all time spent stuck against walls.

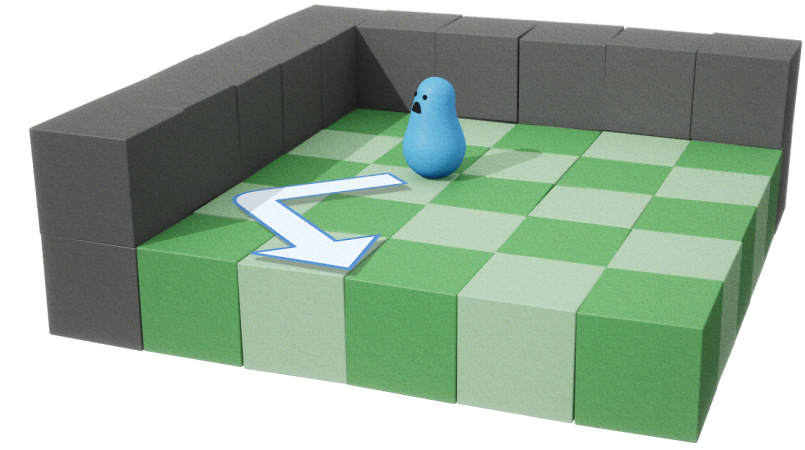


Figure 3.5: Prey Bouncing off Wall
Prey will deflect off of a wall upon impact.

2. **Fleeing**: When a predator becomes aware of a prey, the prey will also become aware of the predator and begin to flee in the direction opposite of the predator. This fleeing is also accompanied by a slight "panic", so the prey does not flee in a perfect line, but rather, it attempts an almost chaotic escape in a direction away from the predator.

Figure 3.6: Prey Fleeing Predator
Prey will flee from the predator when within the predator's sensing zone.

3.2.3 Food Gathering

Food Gathering is essentially the same as Predator Versus Prey, but the prey have a speed of zero. This implementation of pursuit forces the predator to spend more effort exploring the environment instead of waiting for prey to stumble onto it.

In the previous two pursuit implementations, the predator could often remain in a small subsection of the arena and patiently catch the prey that eventually move into it. Sometimes, the predator would not move at all, and would sit idly until a prey moved close enough to be pounced on.

In Food Gathering, the predator must now evolve to effectively catch prey whilst also thoroughly exploring the environment in which it resides. This effectively removes any idling that the predator may have done throughout the previous pursuit scenarios.

3.3 Genetic Programming

The Java-based Evolutionary Computation Research System (ECJ) (version 27) [43] was used as the main system driving this research. ECJ was primarily chosen due to a pre-existing familiarity with the system, but as the powerful and easy-to-use system that it is, it would have been an obvious choice regardless. Most notably, ECJ offered many required features out-of-the-box, including automatic multi-threading capabilities. For detailed information regarding ECJ, see [43].

3.3.1 Strongly-typed Language

Table 3.2: GP Variable Types

TYPE	DESCRIPTION
Condition (C)	Represents a boolean and stores either -1 (false) or +1 (true).
Float (F)	Stores a single floating point value.
Vector (V)	Stores two floating point values representing X and Y.

The GP language is strongly-typed with three variable types: conditions (booleans), floats, and vectors (two-dimensional). With the environment being two-dimensional, vectors are crucial to producing meaningful solutions. Floats were added to assist the GP with manipulating the vectors. Conditions were added to provide straightforward decision making. Table 3.2 lists the variable types used in this language and their corresponding descriptions.

Beyond the basic arithmetic functions, the GP language is packed with a variety of interesting functions to be experimented with. Some functions are duplicated for each language type, but the majority are unique to their corresponding language type.

Each language type has its own *Branch* function in which a condition is evaluated and one of the two children are returned based on the condition. This is essentially an "if" statement. Each language type has *Set* and *Get* functions. The Set function will store the input value in a location outside of the GP tree and the Get function will return that value to the GP tree. Each language type also has

Table 3.3: GP Language Functions - Conditions

FUNCTION	DESCRIPTION
Branch (C::=$C_1C_2C_3$)	Returns C_2 if C_1 is true. Returns C_3 otherwise.
Ephemeral (C)	Returns a random condition. Condition is initialized on node creation.
EqualTo (C::=F_1F_2)	Returns true if F_1 and F_2 are equal (+/- 0.1). Returns false otherwise.
GreaterThan (C::=F_1F_2)	Returns true of F_1 is greater than F_2. Returns false otherwise.
Get (C)	Returns the value stored by "Set".
HittingWall (C)	Returns true if the next step would hit a wall. Returns false otherwise.
ProximityCheck (C)	Returns true if a prey is within the sensing radius. Returns false otherwise.
SeesPrey (C)	Returns true if a prey is within predator's field of view. Returns false otherwise.
SeesWall (C)	Returns true if a wall is within predator's field of view. Returns false otherwise.
Set (C::=C_1)	Returns C_1 and stores the value in memory. Value will be unchanged until set again.
ToCondition (C::=F_1)	Returns true if F_1 is non-negative. Returns false otherwise.

their own *Ephemeral*, a randomly initialized value. For floats, this value is between -1.0 and +1.0, inclusive. Vectors are the same as floats, but with a separate value for X and Y. Conditions round the value to an exact -1.0 or +1.0.

The set of condition functions includes several terminals returning the boolean states set by the predator agent. When the predator sees a prey (prey is within the predator's field of view) the *SightCheck* terminal returns true. Likewise, with a wall and the *SeesWall* terminal. If a prey is within the predator's sensing radius, the *SensesPrey* terminal will return true. If there is a wall directly in front of the predator (the predator is in a cell adjacent to a wall and is facing a wall), *WillCollide* will return true. The entire set of condition functions is described in Table 3.3.

The float functions are all relatively straightforward, with the exception of a few. The *BreakVectorX* and *BreakVectorY* functions are designed to give access

Table 3.4: GP Language Functions - Floats

FUNCTION	DESCRIPTION
Add (F::=F_1F_2)	Returns sum of F_1 and F_2.
Average (F::=F_1F_2)	Returns average of F_1 and F_2.
Branch (F::=$C_1F_1F_2$)	Returns F_1 if C_1 is true. Returns F_2 otherwise.
BreakVectorX (F::=V_1)	Returns X value from vector V_1.
BreakVectorY (F::=V_1)	Returns Y value from vector V_1.
Cosine (F::=F_1)	Returns cosine of F_1.
CurrentTick (F)	Returns the current tick of the simulation.
Divide (F::=F_1F_2)	Returns F_1 divided by F_2. Dividing by zero produces zero.
Dot (F::=V_1V_2)	Returns dot product of V_1 and V_2.
Ephemeral (F)	Returns a random float [-1.0,+1.0]. Float is initialized on node creation.
Get (F)	Returns the value stored by "Set".
Invert (F::=F_1)	Returns F_1 flipped.
MaxXY (F::=F_1F_2)	Returns the greater value among F_1 and F_2.
MinXY (F::=F_1F_2)	Returns the smaller value among F_1 and F_2.
Multiply (F::=F_1F_2)	Returns F_1 multiplied by F_2.
Set (F::=F_1)	Returns F_1 and stores the value in memory. Value will be unchanged until set again.
Sine (F::=F_1)	Returns sine of F_1.
StepsSinceLastCatch (F)	Returns length of time elapsed since last catch.
Subtract (F::=F_1F_2)	Returns F_1 subtracted by F_2.
VectorLength (F::=V)	Returns length of the vector V.

to the individual components of a vector; they simply take a vector and return the corresponding float value. The *CurrentTick* returns the current tick in the simulation, that is, if the simulation has been running for 250 time units, CurrentTick returns 250.

Most of the vector functions are self-explanatory, save for a few. Functions like *PredLocation*, *SightHitResult*, and *SenseHitResult* will all return location vectors set by the predator's percepts. *ReplaceX* and *ReplaceY* are the counterparts to the BreakVectorX and BreakVectorY float functions, as they will *replace* an individual component within a vector.

Table 3.5: GP Language Functions - Vectors

FUNCTION	DESCRIPTION
Add (V::=V_1V_2)	Returns sum of V_1 and V_2.
Branch (V::=$C_1V_1V_2$)	Returns V_1 if C_1 is true. Returns V_2 otherwise.
Ephemeral (V)	Returns a random vector ([-1,1],[-1,1]). Vector is initialized on node creation.
Get (V)	Returns the value stored by "Set".
Inverse (V::=V_1)	Returns the inverse (negated) V_1.
MakeVector (V::=F_1F_2)	Returns vector (F_1,F_2).
Normalize (V::=V_1)	Returns the unit vector of V_1.
PredLocation (V)	Returns predator's location vector.
PredRotation (V)	Returns predator's rotation vector.
ProximityCheck (V)	Returns the vector location of a sensed prey. Returns the predator's location if no prey is sensed.
ReplaceX (V::=F_1V_1)	Returns V_1 with X replaced by F_1.
ReplaceY (V::=F_1V_1)	Returns V_1 with Y replaced by F_1.
Rotate (V::=F_1V_1)	Rotates V_1 by F_1 degrees.
Set (V::=V_1)	Returns V_1 and stores the value in memory.
SightCheck (V)	Returns the vector location of a seen prey. Returns the predator's location if no prey is seen.
Subtract (V::=V_1V_2)	Returns V_1 subtracted by V_2.
SwapXY (V::=V_1)	Returns V_1 with swapped X and Y values.

3.3.2 Parameters

The GP parameters are, for the most part, universal across all experiment series. Table 3.6 lists the most important parameters for all three experiment series.

3.4 Evaluation Strategies

Within this paper, it should be noted that the *fitness* of a solution will always be in reference to the fitness-based evaluation score, while the *diversity* of a solution will always be in reference to the diversity-based evaluation score. The GP will see its version of fitness as the weighted combination of our version of fitness and diversity.

The focus of this research revolves around the balance between fitness-based evaluation and diversity-based evaluation strategies. On the extreme ends, an individual will only be evaluated with one of these strategies, but those experiments

Table 3.6: GP Parameters

PARAMETERS	SERIES A	SERIES B	SERIES C
GENERATIONS	100	100	100
POPULATION SIZE	1000	1000	1000
CROSSOVER RATE	0.90	0.90	0.90
MUTATION RATE	0.10	0.10	0.10
CROSSOVER MAX DEPTH	17	17	17
MUTATION MAX DEPTH	17	17	17
CROSSOVER TRIES	10	10	10
MUTATION TRIES	1	1	1
TOURNAMENT SIZE	3	3	3
ELITISM SIZE	3	3	3
GROW MIN DEPTH	5	5	5
GROW MAX DEPTH	5	5	5
GROW RATE	0.50	0.50	0.50
HALF MIN DEPTH	2	2	2
HALF MAX DEPTH	6	6	6
HALF GROW RATE	0.50	0.50	0.50
TERMINAL RATE	0.10	0.10	0.10
NON-TERMINAL RATE	0.90	0.90	0.90
SIMULATION LENGTH	5000	5000	5000
PREDATOR SIGHT DISTANCE	20	20	20
PREDATOR SENSE DISTANCE	5	5	5
PREDATOR FIELD OF VIEW	90°	90°	90°
PREY AMOUNT	20	20	20
PREY SPEED	0.70	0.70	**0.00**
PREY CAN FLEE	FALSE	**TRUE**	FALSE
PREY CAN BOUNCE	FALSE	**TRUE**	FALSE

are solely for finding baselines for comparison.

Any experiment with a chance at achieving the goals of this research will need a balance between fitness-based evaluation and one of the four diversity-based evaluation strategies. The evaluation strategies used through this research are as follows:

3.4.1 Fitness-based Evaluation

As one would expect, the fitness-based evaluation is the most straightforward strategy used in this research. This is a generic EA strategy, where evolution is solely

guided by fitness. Throughout an individual's simulation, a point is awarded every time a prey is caught. When the simulation ends, the awarded points are normalized and stored as the fitness.

3.4.2 All Neighbours

All Neighbours was the most intuitive approach when developing a diversity-based evaluation strategy. All Neighbours will assign a diversity score to every member of the population based on how unique that individual is compared to the population average. In order to calculate this score, the following steps are followed:

1. **Calculate Individual Characterization Vectors**: The characterization vector for each individual is first calculated. As described in Chapter 2, all of the individual's quantitative behaviours measures are collected and combined into a single vector.

2. **Create Population Average Characterization Vector**: The average characterization vector is used as the baseline for comparing other individuals. This vector is created simply by averaging every individual's characterization vector into a single vector.

3. **Create Individual Distance Vectors**: The distance vector for each individual is calculated for ease of use and has the same number of components as the characterization vector. Each component in an individual's characterization vector is compared against the corresponding component in the population average characterization vector, and this difference is stored in the appropriate slot within the individual's distance vector.

4. **Rank Individual Distance Vectors**: Each individual is finally placed through a sum of ranks evaluation (Section 2.4). Each of the vector components that comprise the individual's distance vector is used as an objective, with a higher distance being better. The final scores returned from the sum of ranks process will be the diversity scores for the individuals.

Once this process is complete and each individual has a diversity score, the final score can be calculated. The fitness-based evaluation score and the diversity-based evaluation score are multiplied by their corresponding weights and then summed together. This final value is the individual's fitness.

3.4.3 All Neighbours with Archive

It was found through empirical analysis that the All Neighbours evaluation strategy would suffer from an oscillating effect. Details of this are explained in Chapter 5.

An archive was added to combat this pitfall. Instead of comparing an individual against the population average characterization vector, an individual is compared against the *archive* average characterization vector. The archive average characterization vector is simply an average of the population average characterization vectors collected from the previous five populations.

With the archive average characterization vector being used in place of the population average characterization vector during individual comparisons, the same steps are followed to calculate the final fitness score for an individual.

3.4.4 K-Nearest Neighbours

K-Nearest Neighbours was often referenced in similar works ([25], [27], [23]) so it was an obvious choice for an additional diversity-evaluation method. Similar to All Neighbours, individual characterization vectors are collected and compared against an average characterization vector. The difference with K-Nearest Neighbours is that the average characterization vector is the average of the characterization vectors collected from the k-nearest neighbours relative to the individual being evaluated. Therefore, the same sum of ranks algorithm is done, as with All Neighbours.

It has been stated that a good value for k is the square root of the population size [44]. With the population size in this research always being 1000, k will always be 32.

3.4.5 K-Nearest Neighbours with Archive

It was often the case that k-nearest neighbours was used with an archive in other work [25], so this research includes the same. The distinction between a vanilla k-nearest neighbours and a k-nearest neighbours with archive approach is maintained to explore the differences.

It was assumed that this strategy would yield the best results considering the popularity it has among related works. This assumption proves to be true in Chapter 5, and as such, continues to be used throughout Chapters 6 and 7 as well.

Chapter 4

Experiment Design

Every experiment was conducted 30 separate times, with the data being averaged among all 30 runs to produce the final results. Baseline experiments never utilize any multi-objectivization or diversity evaluation strategies. Non-baseline experiments will always utilize multi-objectivization and diversity evaluation strategies to some degree.

Specifically, each experiment will use a weighting between fitness-based evaluation and diversity-based evaluation. This weighting will be referred to in the format XF/YD, where X is the percentage of weight allocated toward the fitness score and Y is the percentage of weight allocated toward the diversity score. For example, an experiment labelled with 75F/25D will use $(FITNESS \times 0.75 + DIVERSITY \times 0.25)$ to calculate the final score.

An experiment *Series* will encapsulate all experiments done in a particular implementation of the pursuit domain. An experiment *Set* will refer to a group of four experiments within a series that focus on a particular diversity evaluation strategy, plus the baseline. For example, all experiments in Experiment Series A use the same environment. Experiment Series A also explores four strategies for diversity evaluation, so Experiment Series A will have four sets of experiments, and each set will include Experiment Series A's baseline experiment along with the four diversity-based experiments.

As mentioned previously, the four diversity strategies are as follows: All Neigh-

Table 4.1: Summary of Experiments

SERIES	A	B	C
DESCRIPTION	To determine the best strategy and weighting.	To further examine result from Series A.	To further examine result from Series A.
EXPERIMENTS	**AN 100F/0D**	**KNNA 100F/0D**	**KNNA 100F/0D**
	AN 75F/25D	KNNA 75F/25D	KNNA 75F/25D
	AN 50F/50D	KNNA 50F/50D	KNNA 50F/50D
	AN 25F/75D	KNNA 25F/75D	KNNA 25F/75D
	AN 0F/100D	KNNA 0F/100D	KNNA 0F/100D
	ANA 100F/0D		
	ANA 75F/25D		
	ANA 50F/50D		
	ANA 25F/75D		
	ANA 0F/100D		
	KNN 100F/0D		
	KNN 75F/25D		
	KNN 50F/50D		
	KNN 25F/75D		
	KNN 0F/100D		
	KNNA 100F/0D		
	KNNA 75F/25D		
	KNNA 50F/50D		
	KNNA 25F/75D		
	KNNA 0F/100D		

Bolded experiments are baselines, therefore the diversity strategy has no effect.
All baselines in Series A are the same experiment, just measured four separate times.

bours (AN), All Neighbours with Archive (ANA), K-Nearest Neighbours (KNN), and K-Nearest Neighbours with Archive (KNNA).

In each experiment, the diversity scores and the corresponding behaviour distances are measured and included in the results, regardless of whether they were used in the evaluation of the individuals. Every set of experiments follows the same pattern: a baseline experiment (100F/0D) is conducted, followed by the four diversity-based experiments (75F/25D, 50F/50D, 25F/75D, and 0F/100D). This pattern shows the changes brought on by a weight shift from fitness to diversity, where 100F/0D is entirely fitness-based and 0F/100D is entirely diversity-based. A summary of experiments is shown in Table 4.1.

In every experiment, primary importance is given to the fitness and diversity scores. The fitness score is determined by the amount of prey consumed by the predator. The diversity score is an average of the four behaviour distances: wall

Table 4.2: Qualitative Behaviour Descriptions

BEHAVIOUR	DESCRIPTION
WANDERING	Is moving without an obvious pattern.
SCRAPING	Scrapes along the walls of the arena when moving.
BOUNCING	Bounces off of walls upon impact.
SCANNING	Scans up and down each row or column sequentially.
CIRCLING	Moves in circular motions.
SHAKING	Does not maintain a smooth rotation; field of view shakes.
SPLITTING	Flips between two rotations every tick.
CHASING	Chases a prey.
POUNCING	Chases and immediately catches a prey.
IDLING	Does not move.
STALKING	Chases but intentionally avoids catching the prey.
SPRINKLERING	Rotates slowly in one direction and then quickly in the other.
SLOWING	Gradually slows speed.
CAMPING	Spends majority of time in one location.
AVOIDING	Moves around prey.
FLICKERING	Switches between moving and idling every other tick.
HERDING	Chases prey against a wall and holds it there.

impacts, time near prey, cells visited, and average speed. The term "distances" (d) is used because these are not represented as "actual" (a) values, but rather, they represent the degree of uniqueness for an individual's value for a given behaviour compared against the average of that value among a given set of individuals (see Section 1.1.2). If an individual has a high value for its average speed (d), its average speed (a) is probably either very low or very high, while the average speed (a) of the set of individuals it was compared against would be the opposite. If an individual has a very low value for its average speed (d), its average speed (a) is probably similar to whatever the average speed (a) is for the set of individuals it was compared against. As one would then expect, a value approaching zero for any behaviour distance implies convergence among the population, or at least, convergence among the individuals that were evaluated.

The data collected from each experiment can be divided intro three distinct categories: *population trends*, *quantitative behaviours*, and *qualitative behaviours*. Population trend data provides an overview of the evolutionary performance of the entire population over the 100 generations of evolution. Quantitative behaviour

data provides information regarding behaviours that are programmatically measured from the best solutions of an experiment; average speed, for example. Qualitative behaviours provide information regarding behaviours that are empirically measured from the best solutions of an experiment; stalking and pouncing, for example. "Best solutions" or "best individuals" will always be in reference to the best set of individuals from each of the 30 separate runs for a given experiment. The qualitative analyses process of recording and tallying is somewhat subjective since it must be done through human observation. To mitigate any biased decisions, simulations were watched in a random order.

Throughout the qualitative behaviour analyses, observed behaviours (emergent behaviours) will be marked as either primary or secondary. Table 4.2 lists and describes these behaviours in the order they were observed in. Primary behaviours are behaviours that appear to be the most prominent throughout the simulation. By definition, these behaviours will always be some type of locomotion-based behaviour; wandering, scraping, and scanning for example. Secondary behaviours are observed to be used in tandem with the primary behaviour but are not as prominent. This could include a predator that scrapes the walls but does so with shaky movement; scraping would be primary while shaking would be secondary in this example.

Evaluating agents based on groups of behaviours has been examined in other works and referred to as "personas" [45] [46] [47], but this does not apply in the same way throughout this research. An important note to highlight with respect to the qualitative behaviours in *this* research is not how they are defined or how agent behaviours are categorized into them. Instead, it needs to be understood that these behaviours and their corresponding descriptions are simply observations made by a human after evolution has been completed. The purpose is to provide evidence of an increase in behavioural diversity without requiring the reader to watch each simulation replay on their own.

Chapter 5

Experiment Series A: Predator Versus Prey

Experiment Series A contains the first set of experiments, and as such, was the most broad. Experiment Series A begins with the first baseline experiment, in which the genetic program (GP) is left with full weighting toward the fitness scores and no weighting toward the diversity scores. Following this, four sets of experiments are done, each with their own diversity-based evaluation strategy.

5.1 Baseline

The goal here was to achieve a fitness-based baseline that had room to become both better and worse, so a target fitness-based evaluation score of around 80% was desired. Figure 5.1 shows the population performance trend for this experiment. The diversity score and corresponding quantitative behaviours are included (using all neighbours), although they had no influence on the evolution (the diversity score had a weighting of zero).

After about 70 generations, the GP managed to reach a fitness score of around 80%. This score was then maintained for the remainder of the experiment. Converging on a fitness score of about 80% was desired because it provides opportunity for the addition of diversity-based evaluation techniques to increase it further.

It can be seen in Figure 5.1 that the diversity decreases for AN and ANA as the population converges toward a near-optimal behaviour. This is exactly as one would expect, since a converging population will tend to be comprised of similar or identical individuals. A GP is focused on optimizing fitness, so convergence to comparable behaviours between runs will often arise. The diversity remains relatively unchanged for KNN and KNNA, but considering their comparisons are only made against a small subset of the population, it would be wrong to expect otherwise.

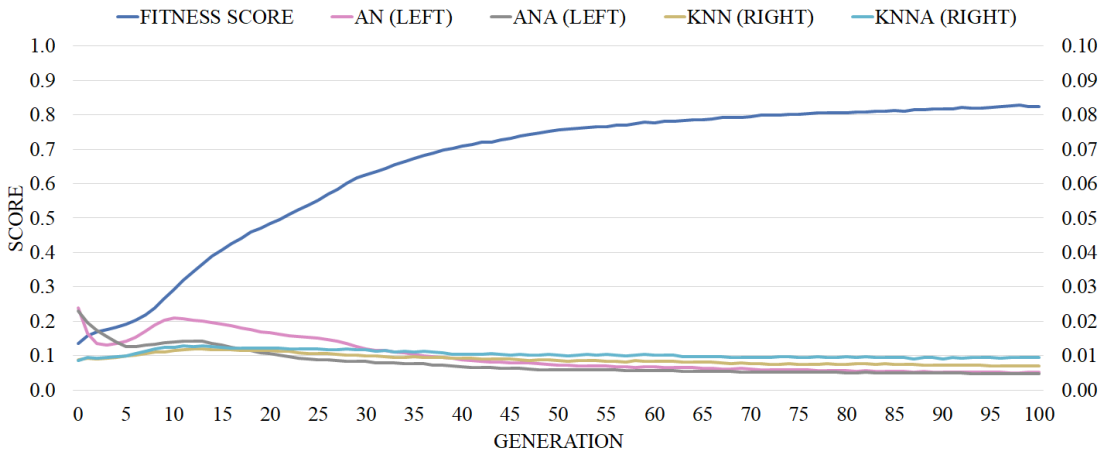


Figure 5.1: Series A - Baseline 100F/0D - Population Trend
Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Diversity scores using AN, ANA, KNN, and KNNA are included for later comparisons.

The baseline experiment was then empirically (qualitatively) analyzed, which required the simulations to be observed by humans, so that emergent behaviours can be recorded and tallied. This type of analysis is important for this research as the subjective qualities of interesting behaviours cannot be easily captured programmatically.

It can be seen in Figure 5.2 that scraping is the primary behaviour throughout all 30 runs. The scraping behaviour (described in Table 4.2), appears to be exploited by the GP due to the unfortunately simple controllers driving the prey agents. Since the prey only add random adjustments to their rotation, they often get stuck on the walls of the arena until a large enough rotation adjustment can be made. From this, the predators evolved to scrape along the walls of the arena

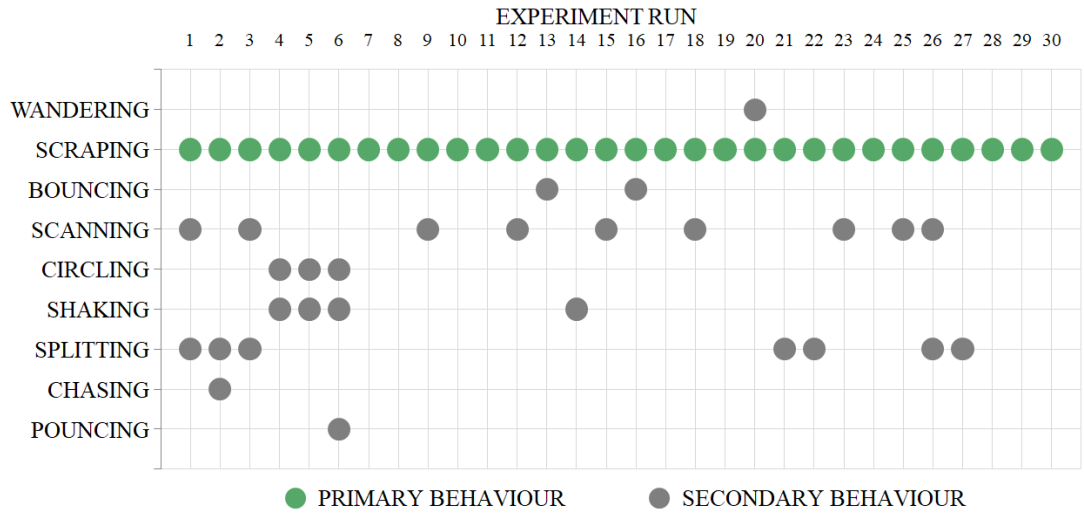


Figure 5.2: Series A - Baseline 100F/0D - Qualitative Behaviours
Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation.

in hopes of finding these easy catches.

This is exactly what should be expected of a purely fitness-based evaluation strategy. As discussed throughout Chapters 1 and 2, fitness-based evaluation is prone to producing the same results between runs. This experiment showed exactly that, with the scraping behaviour being the primary behaviour in all 30 runs.

Other emergent behaviours were also observed but were less prominent than scraping, as seen in Figure 5.2. Scanning was an interesting behaviour and would sometimes occur in tandem with scraping: the predator would scan the walls by scraping back and forth across them. Other behaviours like splitting and shaking rarely had a significant impact on the primary behaviour, as they appeared to simply be a consequence of imperfect vector mathematics evolved by the GP.

Very rarely, the GP would evolve the correct vector mathematics to locate and navigate directly to a prey agent. This resulted in a pouncing-esque behaviour, where the predator would beeline to a prey whenever the prey entered the predator's sensing radius.

The genetic program code generated for the best individual of the first run of this experiment can be seen in Figure E.1.

5.2 All Neighbours

All Neighbours (AN) was the first diversity-based evaluation strategy used in this research. AN was the most intuitive strategy and was the first to come to mind when thinking of techniques to encourage diversity. As stated in Chapter 3, AN compares an individual to every member of the population.

Throughout the set of AN experiments, the final diversity score saw an increase of 26%, while the final fitness score suffered a staggering 75% reduction. However, this is the most extreme comparison to be made (AN 100F/D0 versus AN 0F/100D). This comparison, along with the rest from the AN experiment set, can be seen in Figure 5.3.

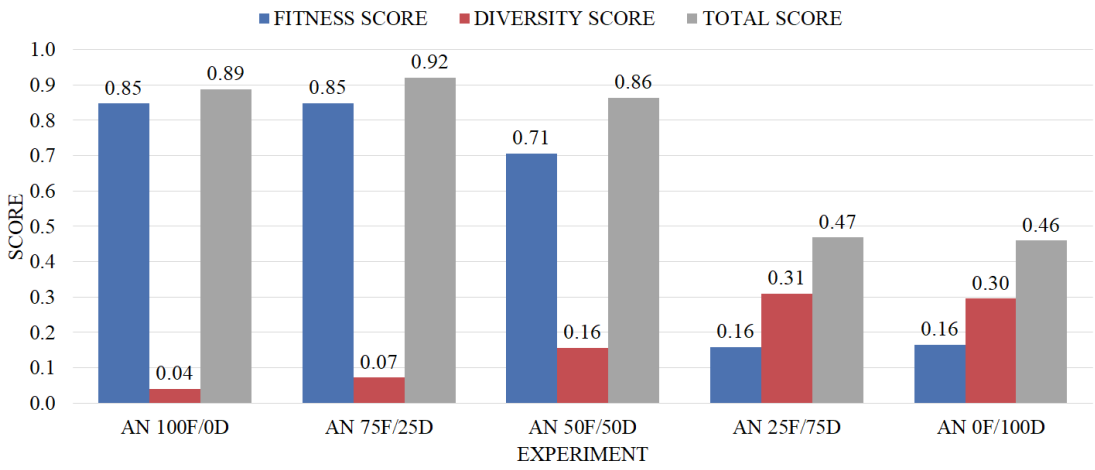


Figure 5.3: Series A - AN - Final Scores
Average scores of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations. Total score is supplementary and was not used by the GP.

This figure shows an obvious trend; as weighting shifts from fitness to diversity, so do the scores. A note of importance, however, is that the total score does not imply superior results. The fitness and diversity scores are not directly proportional in terms of how valuable they are. The nature behind the subjectivity of "interesting" behaviours means even a small increase in diversity may have a significant impact. In other words, a 10% decrease in the fitness score may or may not be worth a 5% increase in the diversity score. With this being understood, the

shift between AN 75F/25D and AN 50F/50D might be worth it, despite the 14% decrease of the fitness score.

It appears that a drastic evolutionary change occurs when the weighting passes the halfway point between fitness and diversity (after AN 50F/50D). Additionally, the diversity score saw no increase when shifting to AN 0F/100D, it actually *decreased* by 3% (while the fitness score remained the same).

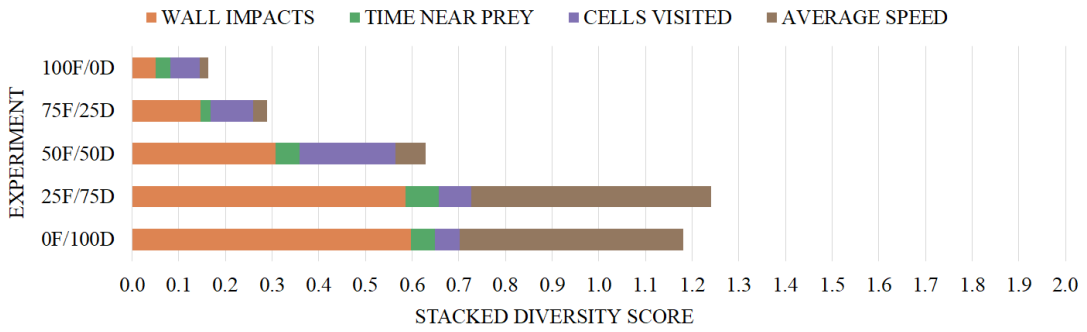
This is contrary to what one would expect, but it appears to make sense upon further examination. For as long as both weightings are not zero, there are five objectives; raw fitness plus the four quantitative behaviours. As soon as the fitness-based evaluation weighting reached zero, the diversity score decreased. This implies that the fitness-based evaluation might be contributing to more exploration throughout the behaviour search space. Considering the goal of this research, entirely removing fitness-based evaluation was never an option, but it is reassuring to see that it actually has a positive impact on achieving diverse behaviours.

The AN 50F/50D experiment produced results which immediately supported the goal of this research. When the fitness and diversity values are weighted evenly, a reasonable balance between scores can be seen. While the final fitness score dropped by 14%, the final diversity score increased by 12%, which is three times its original value of 4%. Such a significant decrease in fitness seems extreme at first, but the gains from the increase in diversity cannot go unnoticed. The predator is still sufficiently capable of catching a majority of the prey, but now, the process with which the predator catches them is much more diverse.

An analysis of the quantitative behaviours further supports this. In Figure 5.4, it can be seen that two particular quantitative behaviours significantly changed throughout the set of AN experiments; wall impacts and average speed. Again, it should be noted that this is not an increase of their *values*, but rather, their *distances*.

Looking at the quantitative behaviour distances for AN 100F/0D (in Figure 5.4), we see all four distances are very low. As discussed earlier, this is to be expected, since the convergence of the fitness-based population would have resulted

Figure 5.4: Series A - AN - Quantitative Behaviours



Average quantitative behaviour distances of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations.

in a lot of similarities among the population. The distances in AN 25F/75D, however, have significantly changed (specifically the wall impacts and average speed). An assumption could be made that ties these two behaviours together: the slower the predator, the less it hits walls, and vice versa. This would explain why both of these behaviour distances rose together, but it is not necessarily the case.

What needs to be understood is that the GP evolved to have these two behaviour distances as the most influential on the diversity of the population. This means that one might expect to see four particular traits in the set of solutions from this experiment: (1) predators that move quickly as they usually would; (2) predators that move slowly despite not benefiting from doing so; (3) predators that are often against a wall; and (4) predators that rarely touch a wall. Again, these are just speculations, but the exciting insight comes from the qualitative behaviour analysis. Performing this analysis on the AN 50F/50D experiment produced an impressive variety of behaviours compared to the baseline (AN 100F/0D), which are outlined in Figure 5.5.

It can be seen here that the scraping behaviour is no longer exploited. Scraping is still the most common primary qualitative behaviour, but there are plenty of others to keep subsequent simulations interesting. This is exactly what was desired; a reasonably effective predator with a variety of behaviours.

While a good portion (around half) of the runs would evolve the typical scraping behaviour found throughout AN 100F/0D, the rest are all vastly different. Most

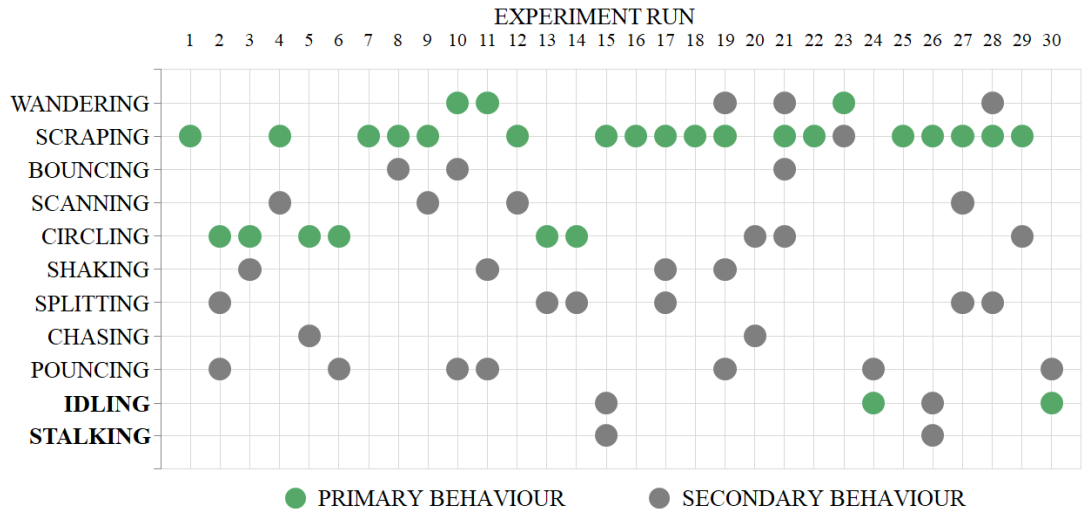


Figure 5.5: Series A - AN 50F/50D - Qualitative Behaviours
Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

notably, two brand new behaviours were introduced; idling and stalking (defined in Table 4.2).

It should be understood that the other nine observed behaviours are not just repeats from AN 100F/0D, as the "implementation" of them is significantly different (to the point of justifying some as new primary behaviours). For example, wandering and circling appear as primary behaviours in AN 50F/50D, but were only ever secondary behaviours throughout AN 100F/0D. Wandering and circling would only be used during small portions of the AN 100F/0D simulations that they were observed in, usually simply as a means of finding a wall to start scraping. In AN 50F/50D, the predator can be observed to spend the majority of some simulations circling, sometimes even doing spirals. There were also a few simulations in which the predator seemed to aimlessly wander throughout the arena. These abnormal behaviours were incredibly interesting to observe, especially considering that they maintained the ability to catch an appreciable amount of prey.

With reference to the high wall impact value in Figure 5.4 (the largest behaviour distance of the AN 50F/50D experiment), scraping contributes to the high values for the actual wall impact counts, while the rest of the primary behaviours con-

tribute to the low values. Wandering, circling, and idling would rarely cause the predator to hit walls, so it makes sense that wall impacts had such a high diversity.

With reference to the cells visited (the second largest behaviour distance of the AN 50F/50D experiment), scraping and idling would result in low values for the actual wall impact counts: scraping because only 2% of the cells in the arena are adjacent to walls, and idling because the predator is not moving at all.

With reference to average speed, idling would obviously produce the lowest actual values, whereas the rest produced a mix of speeds. During the baseline, the predator almost always moved at maximum speed. There was no reason for the predator to move slower, so this is to be expected. During AN 50F/50D, the predator would sometimes go throughout a simulation at a reduced speed, and even waste time getting stuck in corners.

Finally, an analysis on the total score's deviation (Table 5.6) provides more support for the AN 50F/50D experiment. With minimal decrease to the total score, a noticeably large deviation margin can be seen. Typically, a large deviation on the experiment's solutions implies a failure to converge in a consistent manner, but in this research, this is precisely what is desired. Consistent convergence is not interesting, and therefore, a low deviation among solutions is certainly not interesting either.

In short, increasing the diversity weighting with the AN diversity-based eval-

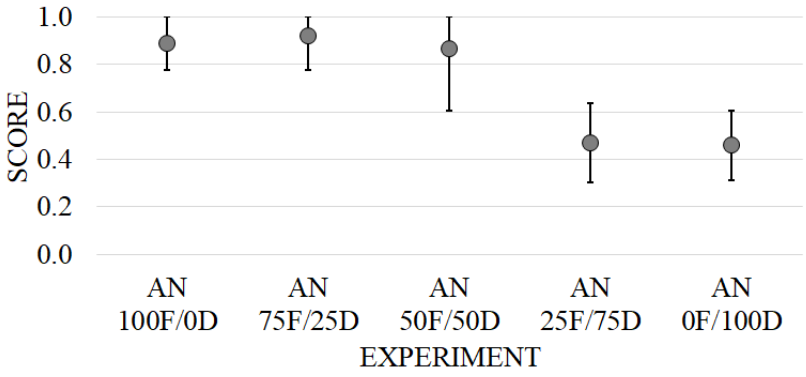


Figure 5.6: Series A - AN - Deviation
Average total score with standard deviation of the best individual from each of the 30 runs used in each experiment.

uation strategy caused a range of new and interesting behaviours to emerge. The final score of AN 50F/50D implies a reasonable level of diversity with an appreciable level of efficacy. The quantitative behaviour analysis of the AN experiment set showed a noteworthy increase of quantitative behaviour diversity beginning at AN 50F/50D. This was then followed with a qualitative behaviour analysis which showed precisely what was desired: a variety of interesting behaviours with an acceptable loss to the fitness score.

5.3 All Neighbours with Archive

All Neighbours with Archive (ANA) was the next diversity-based evaluation strategy to be created. The motivation behind this strategy came from two reasons in particular: (1) to mitigate a saw-toothing effect found in AN, and (2) to follow related works regarding Novelty Search more closely.

As mentioned in Chapter 3, an oscillating effect was observed throughout individual AN experiment runs; this can be seen in Figure 5.7. This particular pattern is alarming to see. It appears that the GP had found two local optima to flip-flop between, and so it became clear at this point that a strategy involving a comparison against every member of the population might be too extreme. From this observation paired with an appreciation for the AN experiment set's successful results, ANA was created.

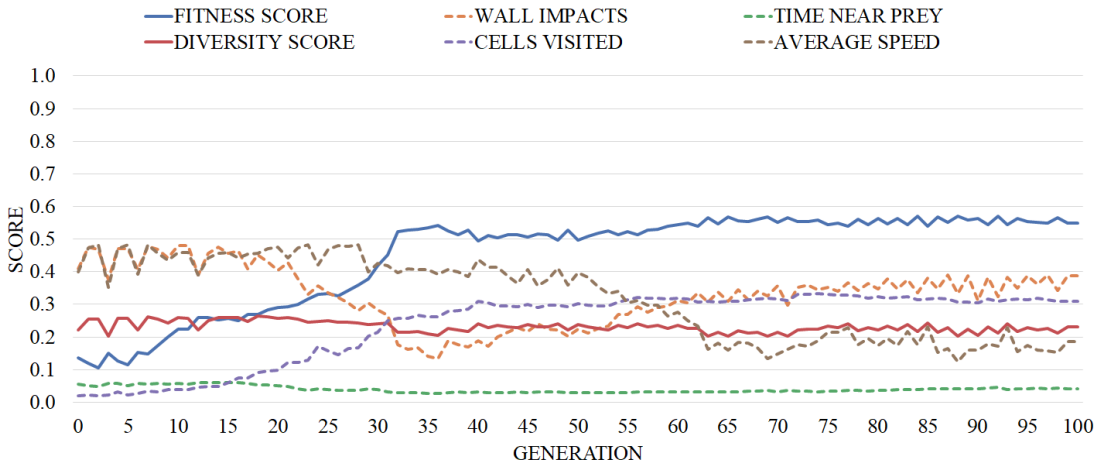


Figure 5.7: Series A - AN 50F/50D - Population Trend
Population trend of 1000 individuals over 100 generations, ***single run only****.*
Solid lines represent scores. Dashed lines represent behaviour distances.

Instead of reducing the set of individuals used during comparison, an archive is added to *expand* it (by four times, specifically). Now, instead of comparing against the entirety of the current generation, the GP compares individuals against the entirety of the current generation *and* the entirety of the previous four generations (5000 comparisons total).

The ANA 50F/50D population trend (for one run) can be seen in Figure 5.8,

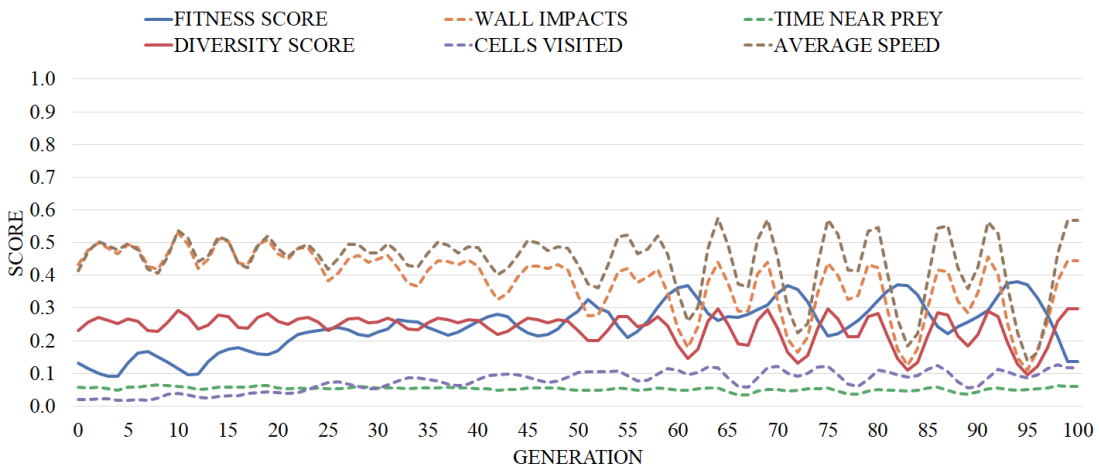


Figure 5.8: Series A - ANA 50F/50D - Population Trend
Population trend of 1000 individuals over 100 generations, ***single run only****.*
Solid lines represent scores. Dashed lines represent behaviour distances.
Take notice of how fitness and diversity oppose each other.

and unfortunately, the oscillating effect only becomes more prominent (lower frequency and greater amplitude). Nevertheless, the results of the ANA experiments were just as impressive as the AN experiments.

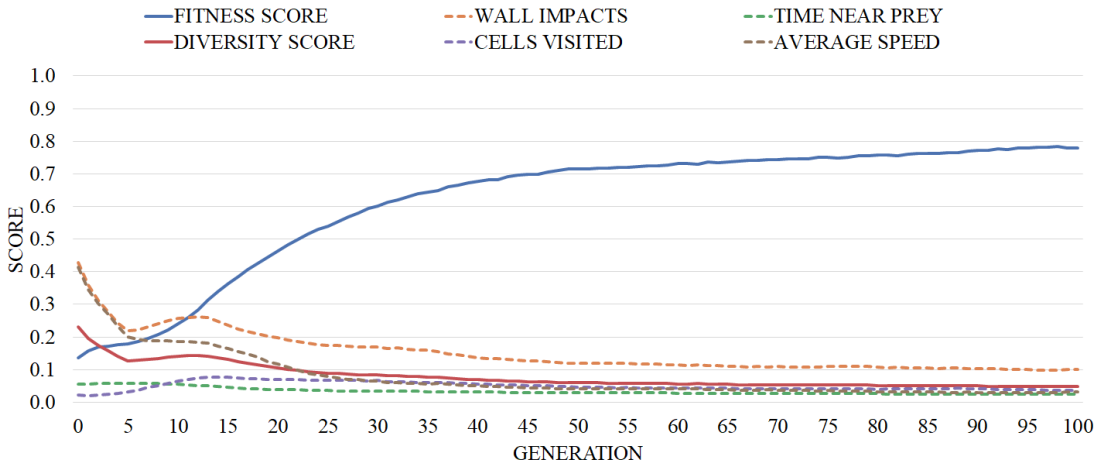


Figure 5.9: Series A - ANA 100F/0D - Population Trend
Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs.
Solid lines represent scores. Dashed lines represent behaviour distances.

To start, the population trend for the baseline experiment (initially shown in Figure 5.1) was expanded to show the individual quantitative behaviour distances. This can be seen in Figure 5.9. As discussed earlier, it makes sense to see the diversity score (and the quantitative behaviour distances) decreasing as the gener-

ations progress. The convergence of a fitness-based population will usually lead to this type of trend. Unfortunately, Figure 5.10 shows that the diversity score *still* fails to increase as the generations progress.

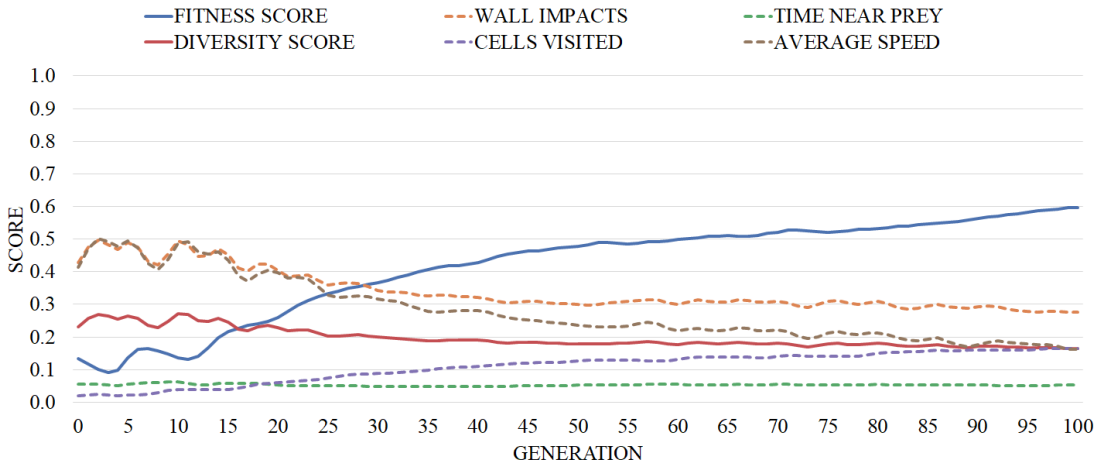


Figure 5.10: Series A - ANA - 50F/50D - Population Trend
Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

However, while the diversity score is decreasing, it can be seen that not all of the quantitative behaviour distances suffer the same fate. Wall impacts and average speed distances fall while cells visited *increases* (and time near prey shows little change). This lack of convergence for three of the four quantitative behaviour measurements suggests opportunity for more interesting results past the 100 generation limit. Future work may benefit from increasing this limit.

Taking a look at the final scores for the ANA experiment set (Figure 5.11), we see little change compared to the AN final scores. ANA 0F/100D managed to achieve the worst fitness score in this research so far, but barring that, ANA's final scores appear to provide no new insights.

The quantitative behaviour analysis, Figure 5.12, fails to provide any new insights either. There are noticeable shifts in the quantitative behaviour distance ratios, but nothing significant enough to warrant further examination. Just like the AN experiment set, the diversity-heavy weightings result in a high distance for wall impacts and average speed.

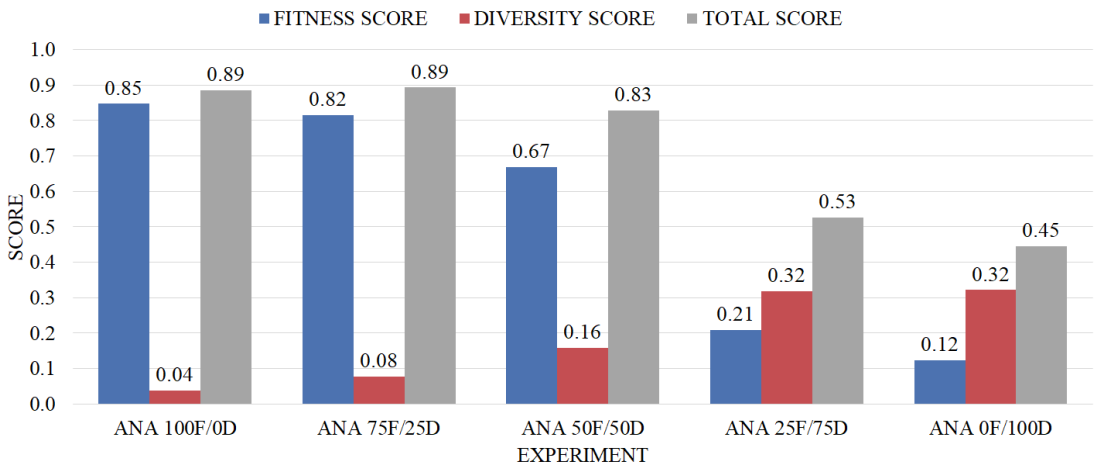


Figure 5.11: Series A - ANA - Final Scores

Average scores of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations. Total score is supplementary and was not used by the GP.

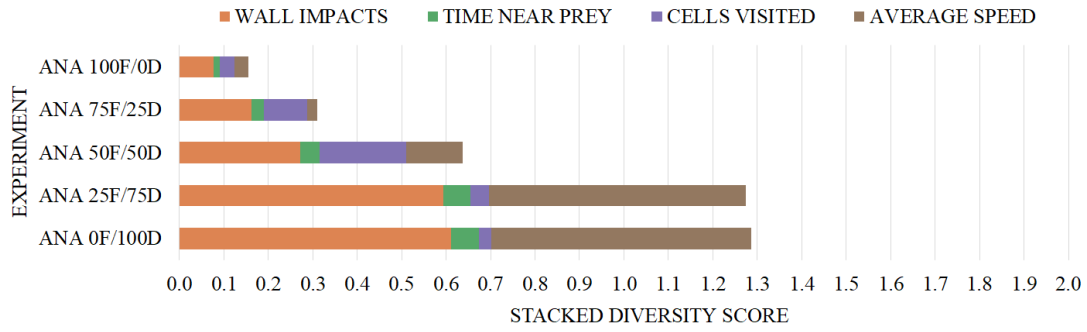


Figure 5.12: Series A - ANA - Quantitative Behaviours

Average quantitative behaviour distances of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations.

Moving on to the qualitative analysis, Figure 5.13 shows the behaviours observed in ANA 50F/50D. Here, we can see the typical favouring toward the scraping behaviour, but with two new behaviours (along with the rest) to accompany it. As with AN, idling was introduced when increasing the diversity score. Stalking, however, did not appear, and was replaced with *slowing*.

The ANA 0F/100D qualitative analysis (Figure 5.14) was a disappointing experience. Almost every predator would sit idly for the entirety of the simulation. Remarkably, the predator still managed to catch *some* prey, despite never moving. With ANA 0F/100D's 12% fitness score (shown in Figure 5.11), the GP appears

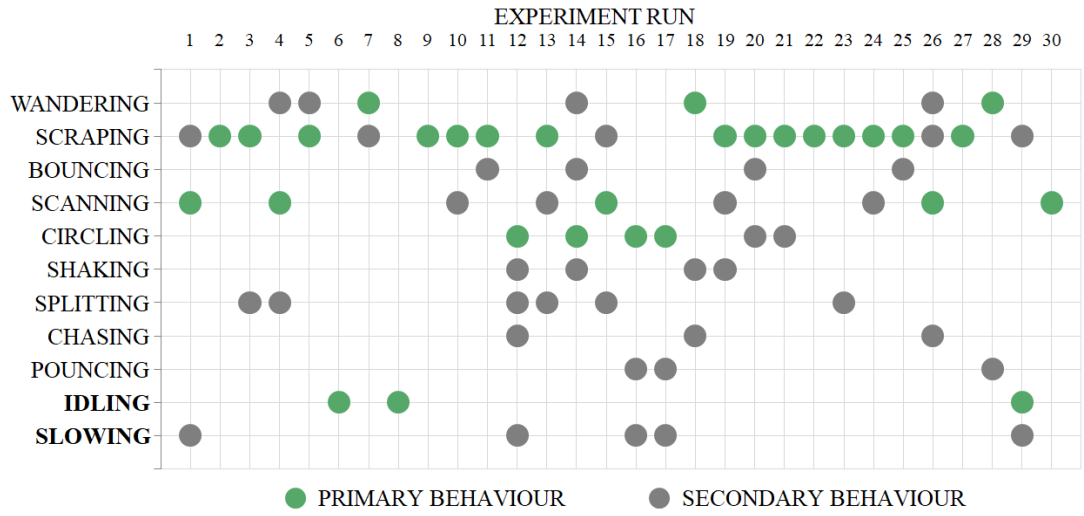


Figure 5.13: Series A - ANA 50F/50D - Qualitative Behaviours
Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

to have evolved to catch approximately 2.4 prey per simulation. This obviously is not true, rather, the GP evolved to do nothing and the simple prey controllers would guide some unfortunate prey directly into the predator. This problem is a consequence of providing the prey controllers with no sensors for perception, and will be remedied in the following experiment series (Chapter 6).

Lastly, an examination of the deviation margins through the ANA experiment series shows a similar result to what was found throughout the AN experiment series. Once again, 50F/50D provides the greatest deviation margin. This time,

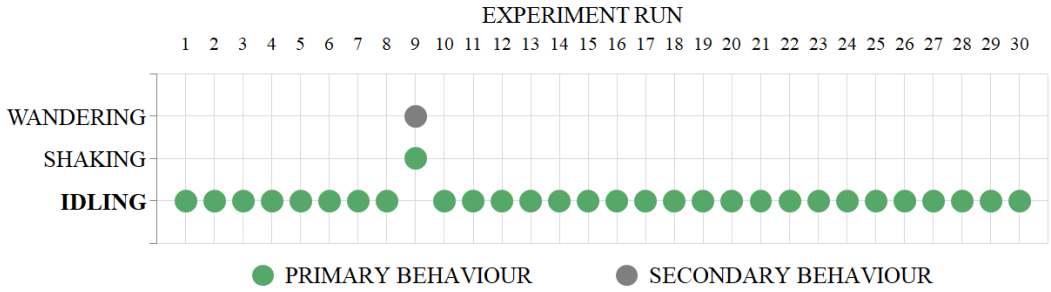


Figure 5.14: Series A - ANA 0F/100D - Qualitative Behaviours
Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

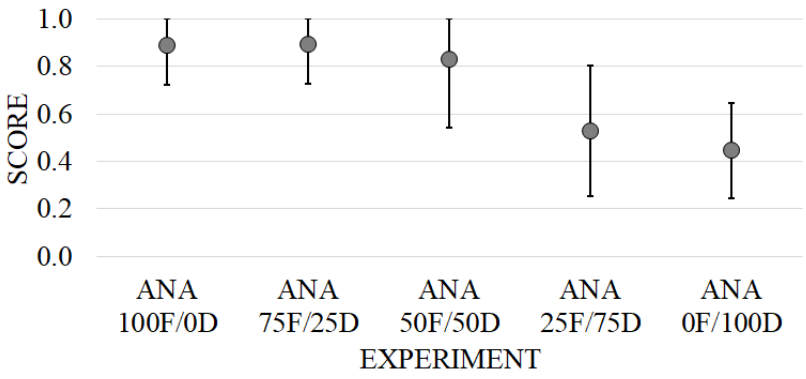


Figure 5.15: Series A - ANA - Deviation
Average total score with standard deviation of the best individual from each of the 30 runs used in each experiment.

however, 25F/75D provided a comparable deviation margin. Unfortunately, the score for ANA 25F/75D was too low, so ANA 50F/50D (like AN 50F/50D) is the best of the set.

5.4 K-Nearest Neighbours

To start, a quick look at the KNN 100F/0D population trend reveals an interesting population movement. This can be seen in Figure 5.16. While AN and ANA would show an immediate deterioration in the diversity score (during the 100F/0D experiments), KNN manages to hold relatively steady. It decreases, of course, as the population converges, but the degree to which it decreases is much less than what is seen in the population trends of AN and ANA.

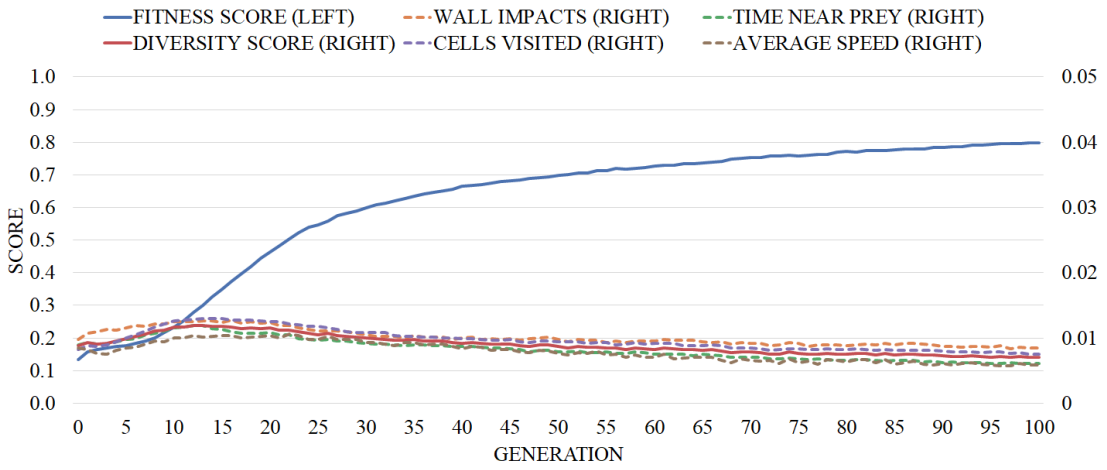


Figure 5.16: Series A - KNN 100F/0D - Population Trend
Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

As a reminder, this is still the same baseline experiment as AN and ANA (Sections 5.2 and 5.3). The diversity score displayed in Figure 5.16 only represents the KNN measurement of diversity throughout that experiment. It should also be noted that the scale for the diversity score has narrowed from a 0-1 range down to a 0-0.05 range. This is simply because the score only has 32 individuals to be built off of, as opposed to the normal 1000 in AN and ANA.

While previous diversity-based evaluation strategies would show deteriorating or stagnating diversity scores, the KNN 50F/50D population trend (Figure 5.17) shows the first instance of a diversity score *increasing* throughout the generations. After a quick jump until generation 20, the diversity score appears to increase at a steady rate for the remainder of the experiment. This is incredibly promising,

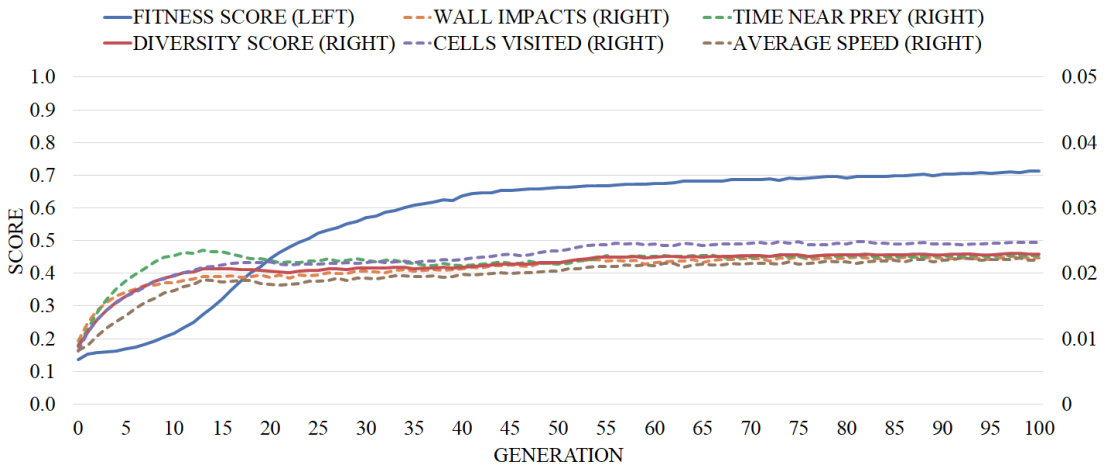


Figure 5.17: Series A - KNN 50F/50D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

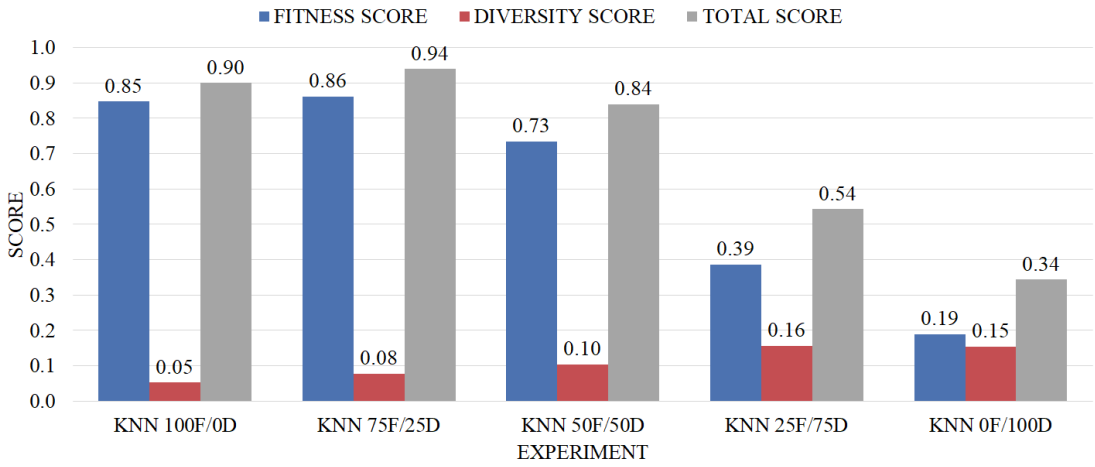


Figure 5.18: Series A - KNN - Final Scores

Average scores of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations. Total score is supplementary and was not used by the GP.

especially considering the fitness score has converged to an appreciable 73%. These values are more accurately displayed in Figure 5.18.

It seemed as though other strategies were bound to a "tug-of-war" when it came to balancing the weights, but KNN appears to benefit in some way when diversity is first introduced. Between KNN 100F/0D and KNN 75F/25D, the fitness score *increases* by 1%. Such a small increase provides no support for any claims (as it could easily fall within the margins of deviation), but it is interesting to see

nonetheless.

The diversity scores have a slow increase and only reach 10% by the 50F/50D weighting. This is disappointing compared to the 16% seen in AN and ANA, but it is not necessarily an indication of less interesting behaviours. To begin exploring the behaviours, we will once again start with the quantitative behaviour analysis in Figure 5.19.

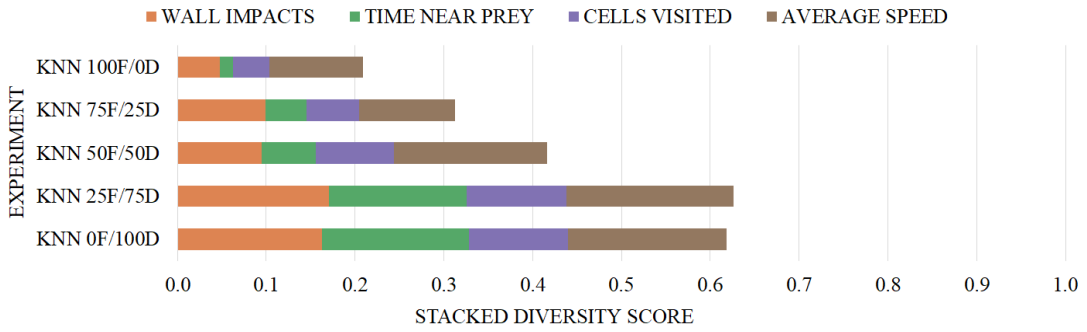


Figure 5.19: Series A - KNN - Quantitative Behaviours

Average quantitative behaviour distances of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations.

The quantitative behaviour distance ratios in KNN are the most evenly distributed distances seen so far throughout this research, especially in KNN 25F/75D and KNN 0F/100D. In previous experiments, the GP would typically neglect the time near prey and cells visited, and spend more effort producing varied behaviours based on wall impacts and average speed.

With time near prey and cells visited being utilized, some new expectations can be made. It might be assumed that the GP will produce behaviours based on the following four generalized behaviours: (1) catching prey the moment they are within sensing radius, (2) avoiding catching prey despite being within sensing radius, (3) exploring the environment with complex and thorough movements, and (4) recycling old paths to avoid exploration.

Some interesting correlations might be speculated as well. For example, time near prey and cells visited could be associated. Since the prey move randomly, following them would result in a lot of cells being visited, whereas simply catching them (which would most likely be done through wall scraping) lacks the increase in

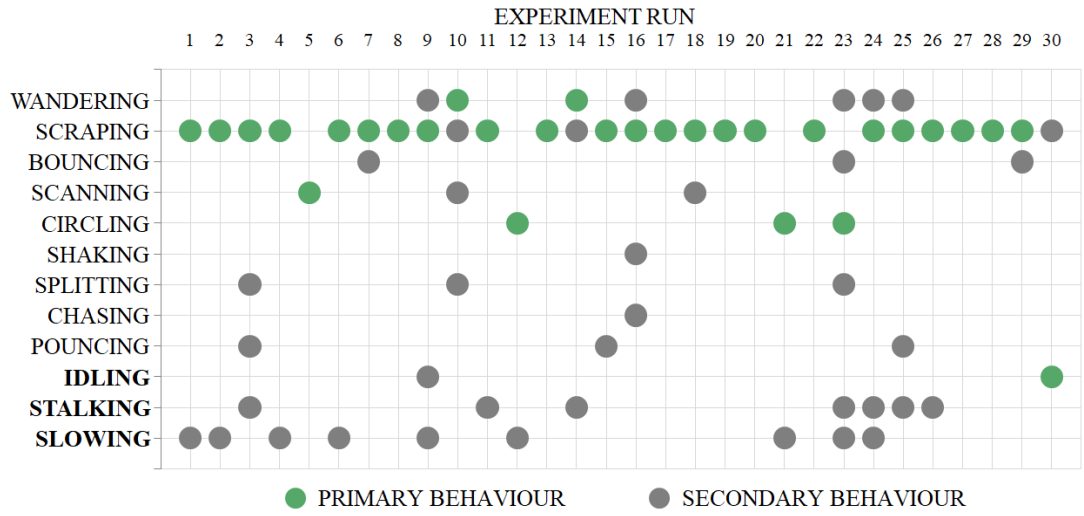


Figure 5.20: Series A - KNN 50F/50D - Qualitative Behaviours
Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

exploration. Again, these are just speculations, as the qualitative analysis in Figure 5.20 will outline what actually happened throughout the solution simulations.

Once again, wall scraping is the favoured behaviour. This is accompanied by the normal set of behaviours, as well as three new behaviours (remember that *new* is with respect to the current experiment set). Idling, stalking, and slowing have been added to the predator's behaviour list, and they have helped produce many interesting solutions.

During KNN 50F/50D, the predator would move at varying speeds throughout the simulations. Sometimes the predator would scrape the left walls slowly and the right walls quickly. Sometimes the predator would move in a spiral fashion by gradually changing the speed whilst maintaining a constant rotation rate. The importance here is the intentional speed decrease on the GP's behalf, sometimes used to create complex movement styles, and sometimes used to simply increase the diversity. It is reassuring to witness these behaviours that exist solely to increase diversity but do not harshly hinder the predator's ability to catch prey.

Unlike the AN and ANA experiment sets, KNN's 0F/100D weighting had an appreciable level of diversity throughout the solutions. While Figure 5.14 dis-

played the opposite effect to what was desired, Figure 5.21 displays a diverse set of behaviours. The fitness score is too low to be considered a good weighting, but it shows that the reduction from 1000 comparisons down to 32 comparisons (of the nearest individuals) has an positive impact on maintaining the diversity of the solutions throughout the diversity weighting increments.

A completely new behaviour, however, did emerge from KNN 0F/100D: cornering. The behaviour's name is the shortened version of "corner camping", a popular strategy used throughout many combat-based video-games. Corner camping is when the player picks a strategic location (some safe corner with good sight-lines) and remains in this location for extended periods of time, waiting to attack other characters until they approach.

Seeing this behaviour emerge through the GP's evolution is an incredibly satisfying achievement. While Grossi's work [7] displayed a GP evolving behaviours used by computer-controlled characters in popular video-games, this experiment displays a GP evolving behaviours used by *human*-controlled characters in popular video-games.

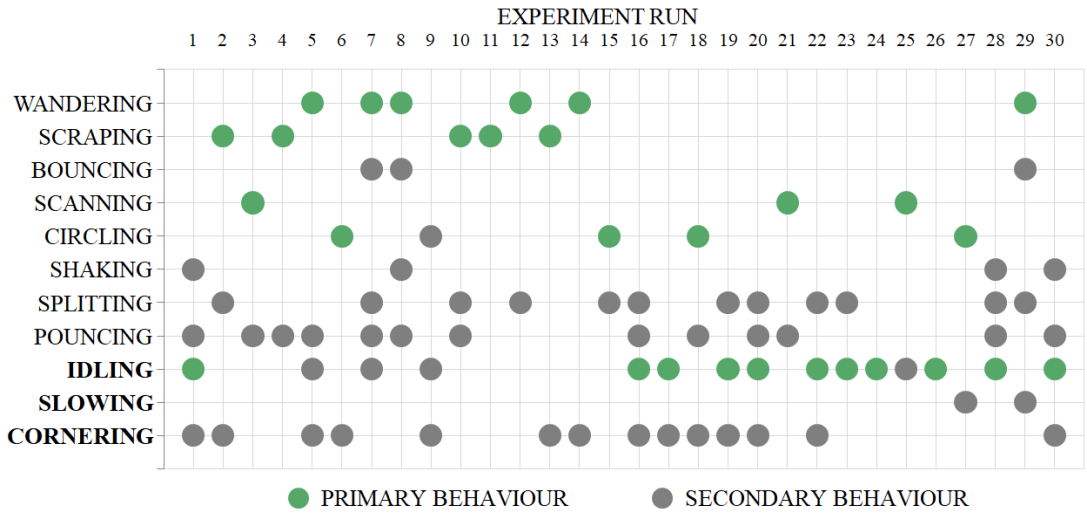


Figure 5.21: Series A - KNN 0F/100D - Qualitative Behaviours
Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

Lastly, a look at the deviation margins for completion's sake; Figure 5.22. It appears KNN 25F/75D had the greatest deviation margins, and therefore, the most diversity. This was also supported in Figure 5.19. Due to the generous amount of time required to complete a qualitative analysis, KNN 25F/75D's qualitative behaviour analysis had to be skipped, but future work may include a more extensive qualitative analysis; perhaps on *all* weightings.

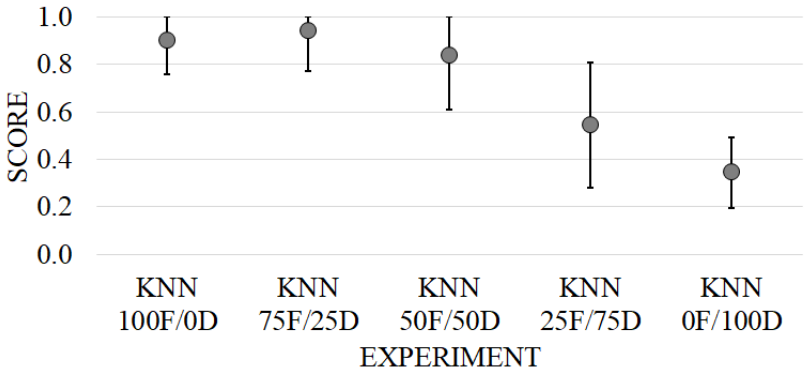


Figure 5.22: Series A - KNN - Deviation
Average total score with standard deviation of the best individual from each of the 30 runs used in each experiment.

5.5 K-Nearest Neighbours with Archive

Lastly, the K-Nearest Neighbours with Archive (KNNA) diversity-based evaluation strategy was implemented. As with most related works, only the k-nearest individuals to the individual being evaluated are used for comparison (k being about the square root of the population size; 32 in these experiments), and an archive being maintained to ensure previous individuals also have an influence. Like the ANA experiments, an archive size of four (plus the current generation) was used. This experiment produced the most impressive results, and as such, becomes the strategy used in two supplementary series of experiments found in Chapters 6 and 7.

To begin, the population trend for the KNNA 100F/0D (Figure 5.23) behaves similarly to the KNN 100F/0D population trend (Figure 5.16), in that the diversity score remains relatively unchanged throughout the generations. As mentioned earlier, it would be expected that the diversity score decreases significantly as the population converges, but when comparing against only the k-nearest neighbours, that pattern is not as pronounced.

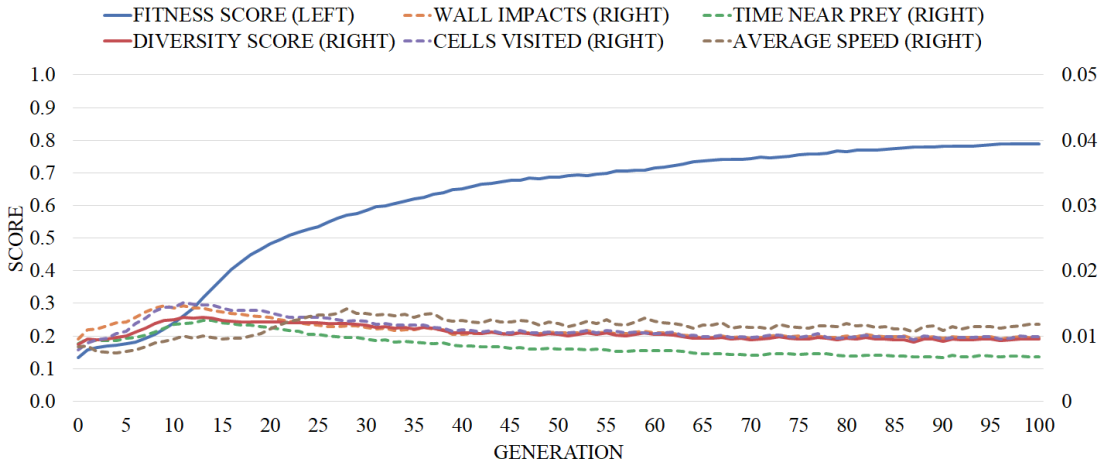


Figure 5.23: Series A - KNNA 100F/0D - Population Trend
Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

The first major improvement that can be seen begins in the KNNA 50F/50D population trend (Figure 5.24) where the population appears to break out of a

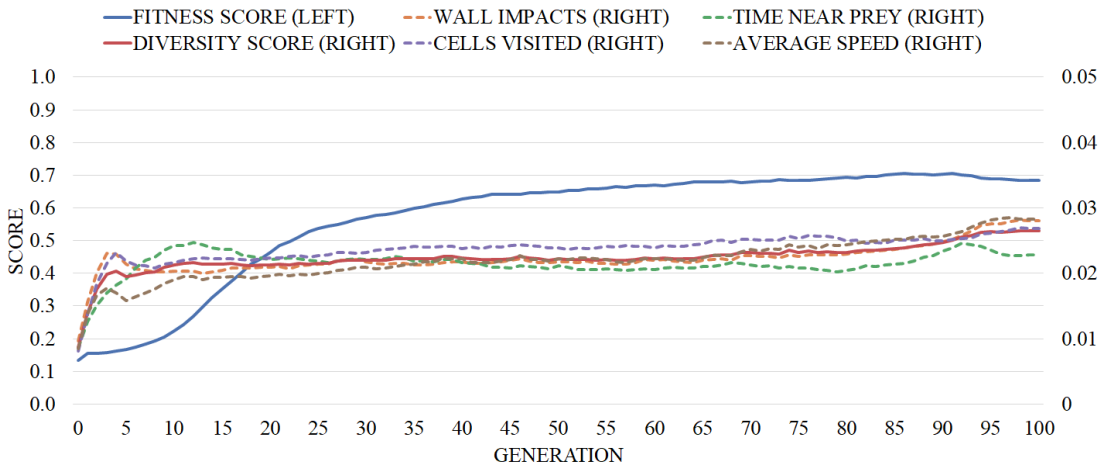


Figure 5.24: Series A - KNNA 50F/50D - Population Trend
Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

local optima. To recall, the purpose of adding the first archive to AN (making ANA) was in an attempt to break out of the local optima that AN's populations would flip between. While it did not work in ANA, it has definitely shown success in KNNA. This breakout of the local optima is more evident within the KNNA 0F/100D population trend, shown in Figure 5.25.

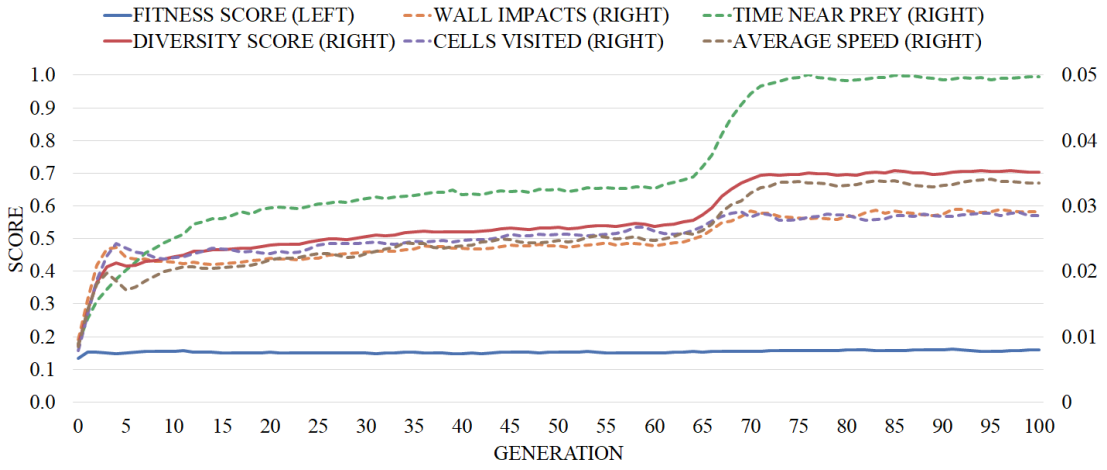


Figure 5.25: Series A - KNNA 0F/100D - Population Trend
Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

This local optima breakout is promising to see, and may continue to improve the diversity score past the 100 generation limit used throughout this research.

Unfortunately, that kind of exploration will have to wait, as discussed in section 8.2.

Next, a look at the final scores for KNNA (Figure 5.26) shows an incredible feat: the fitness score only drops by 3% in KNNA 50F/50D. With the diversity score reaching 16% (equal with the current bests for a 50F/50D experiment), a 3% decrease in fitness is an incredibly impressive trade. For comparison, AN, ANA, and KNN had to sacrifice 20%, 19%, and 47% (respectively) from their fitness scores in order to achieve a diversity score of 16%. In addition to this, the total score for the KNNA 50F/50D experiment is the highest found throughout this research, reaching a value of 98%.

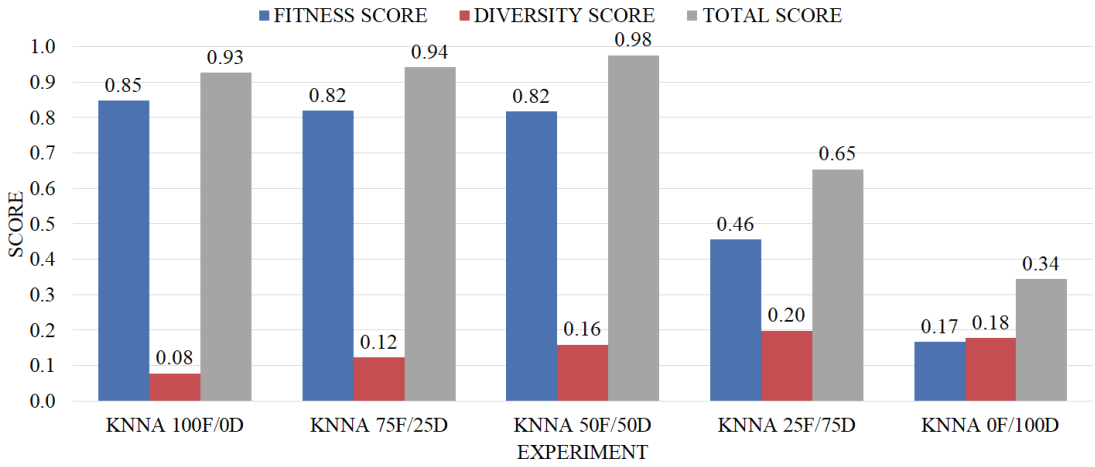


Figure 5.26: Series A - KNNA - Final Scores
Average scores of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations. Total score is supplementary and was not used by the GP.

At this point, the KNNA 50F/50D experiment appears to be the best diversity-based evaluation strategy and weight combination, but this cannot be confirmed until a quantitative and qualitative analysis of the behaviours is done. The diversity score may be high, but the behaviours are not necessarily interesting.

Starting with the quantitative analysis, Figure 5.27, it can be seen that the quantitative behaviour distances double between KNNA 100F/0D and KNNA 50F/50D. This seems unimpressive compared to experiments like ANA 50F/50D, in which the quantitative behaviour distances approximately quadrupled, but again,

this is a matter of comparing against 32 individuals versus 1000 individuals. A more appropriate comparison for quantitative behaviour distances would be between KNN and KNNA, in which KNNA slightly outperforms KNN by achieving a slightly larger stacked diversity score increase.

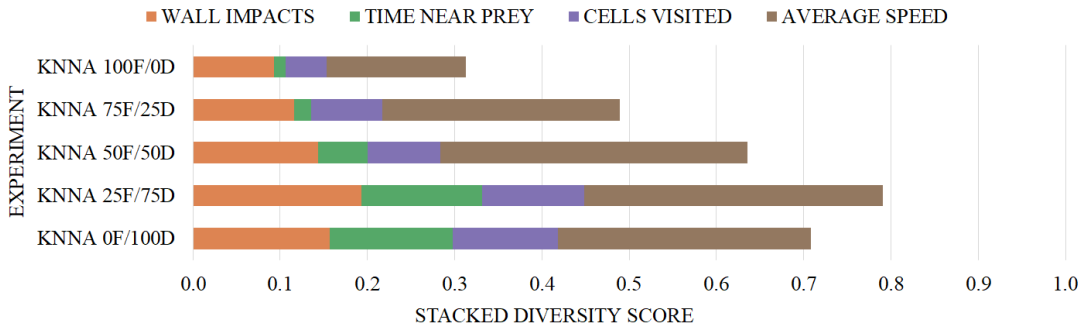


Figure 5.27: Series A - KNNA - Quantitative Behaviours
Average quantitative behaviour distances of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations.

Regardless, the quantitative behaviour analysis cannot give much insight to the emergent behaviours. It shows us that wall impacts and average speed are often used to achieve the high diversity score, but the qualitative analysis is needed to reveal the existence of interesting behaviours. Starting with KNNA 50F/50D, the qualitative behaviour analysis can be seen in Figure 5.28.

This analysis was the most interesting (and most entertaining) to be done thus far. As with the previous experiments, scraping is still commonly utilized as the most effective prey-catching strategy, but a plethora of new and interesting behaviours emerged to accompany it. Almost every simulation from the KNNA 50F/50D experiment utilized scraping, but a high diversity score is maintained with the existence of many other behaviours in an effective manner.

Not only did five new behaviours emerge, but many combinations of existing (already identified) behaviours also emerged to create much more high-level emergent behaviours. The following are just a few examples that stood out in particularly interesting ways:

1. **Sleeping To Scraping**: The predator would sit idly and ignore any prey crossing its field of view. Once a prey crossed the predator's sensing radius,

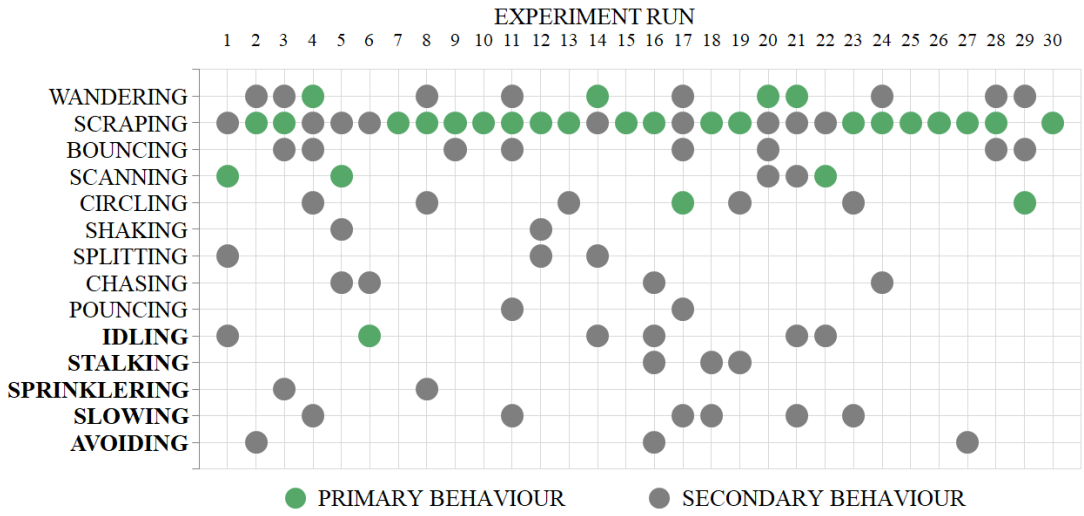


Figure 5.28: Series A - KNNA 50F/50D - Qualitative Behaviours
Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

it would chase the prey to a wall and begin scraping for the remainder of the simulation.

2. **Strolling To Scraping**: Similar to above, but the predator would move at an especially slow speed in a circular motion until hitting a wall. Then the predator would begin the normal scraping behaviour.

3. **Selective Bouncing**: The predator would scrape along three walls as normal, but bounce along the fourth. Sometimes during the bouncing, the predator would pounce on nearby prey before returning to the wall. This is especially interesting since the pouncing behaviour requires the GP to evolve the correct vector mathematics to locate and navigate to detected prey, but scraping has no need for such mathematics.

4. **Passive Wall Scraping**: The predator would scrape the walls as normal, but would stop and wait for stuck prey to move out of the way before continuing forward. This would sometimes occur at seemingly random times, giving the impression that the predator is a "picky eater" and was waiting for a different catch.

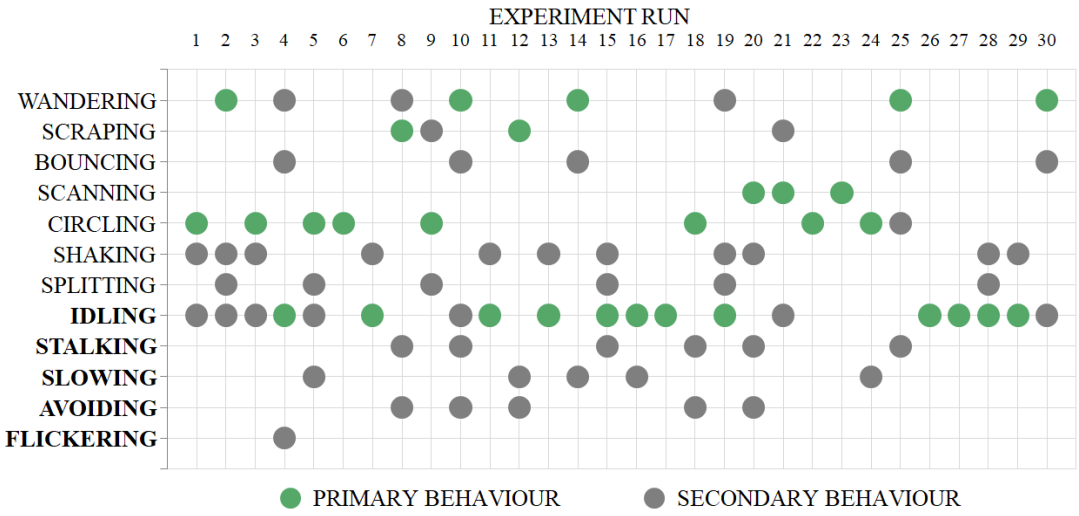


Figure 5.29: Series A - KNNA 0F/100D - Qualitative Behaviours
Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

5. **Scraping With Chasing**: The predator would scrape the walls as normal, but if a prey got unstuck and left the wall moments before capture, the predator would chase and catch the prey. After catching, the predator would turn around and return to scraping.

Ultimately, it seems as though the GP had evolved to switch between different behaviours throughout the simulation in order to balance the fitness score with the diversity score. In most of the KNNA 50F/50D experiments, the predator

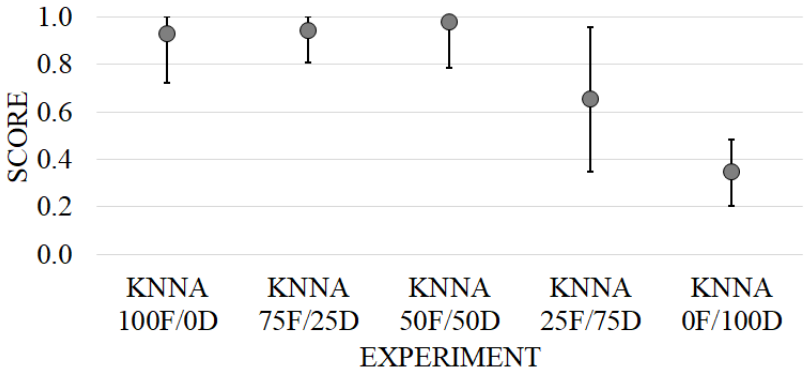


Figure 5.30: Series A - KNNA - Deviation
Average total score with standard deviation of the best individual from each of the 30 runs used in each experiment.

would efficiently scrape the walls to catch as many prey as possible, but in order to maintain a high diversity score, abnormal behaviours were sprinkled throughout the simulation. This, without a doubt, is the emergence of interesting behaviour.

The genetic program code generated for the best individual of the first run of KNNA 50F/50D can be seen in Figures E.2 and E.3.

5.6 Series A Discussion

Experiment Series A contained a vast amount of data. After thoroughly examining the quantitative and qualitative analyses, it appears that KNNA provides the best results. Specifically, the 50F/50D weighting with KNNA diversity-based evaluation provided the most interesting results whilst maintaining an appreciable level of efficacy. Before making any final conclusions on Experiment Series A, a collection of statistical analyses needs to be done.

To reiterate the goal of this research, an evaluation strategy needs to be found with which diverse and effective behaviours can emerge. It appears KNNA 50F/50D is an excellent example of this, but it would be more impactful if it could be proven statistically. Using non-parametric statistical analyses on the sets of solutions, statistical significance can be measured between each of the diversity-based evaluation strategies and their five weightings. Specifically, 380 Mann–Whitney U tests need to be done to cover every combination of diversity-based evaluation strategies and their five weightings; 190 for fitness score comparisons, and 190 for diversity score comparisons. The fitness score comparisons are shown through Tables 5.1, 5.2, and 5.3. The diversity score comparisons are shown through Tables 5.4, 5.5, and 5.6.

The 380 Mann–Whitney U tests adhere to the following interpretations [48]:

1. The values among both sets are all independent of one another; an adjustment to one value would not affect any other values.

2. The values used throughout the two sets are ordinary; they can be directly compared to determine which is superior.

3. The null hypothesis indicates that the difference between the two sets is not statistically significant.

4. The alternative hypothesis indicates that the difference between the two sets *is* statistically significant.

5. Rejecting the null hypothesis when comparing two sets of fitness scores implies that the fitness scores did not suffer in any significant way.

6. Failing to reject the null hypothesis when comparing two sets of diversity scores implies that the diversity scores improved in some significant way (for one of the sets).

During these tests, if a particular experiment can be shown to have no significant difference among the fitness scores, but *with* a statistical significance among the diversity scores, then that test achieves the goal of this research (statistically). With this in mind, the results from the fitness-based and diversity-based Mann–Whitney U tests are combined in Tables 5.7, 5.8, and 5.9 to represent instances of success.

Table 5.1: Series A - Mann–Whitney U Tests on Fitness - Part 1

	AN					ANA				
	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D
AN 100F/0D	-	0.416	0.001	0.000	0.000	-	0.149	0.000	0.000	0.000
AN 75F/25D	-	-	0.002	0.000	0.000	0.416	0.261	0.000	0.000	0.000
AN 50F/50D	-	-	-	0.000	0.000	0.001	0.022	0.469	0.000	0.000
AN 25F/25D	-	-	-	-	0.386	0.000	0.000	0.000	0.198	0.279
AN 0F/100D	-	-	-	-	-	0.000	0.000	0.000	0.625	0.022
ANA 100F/0D	-	-	-	-	-	-	0.149	0.000	0.000	0.000
ANA 75F/25D	-	-	-	-	-	-	-	0.001	0.000	0.000
ANA 50F/50D	-	-	-	-	-	-	-	-	0.000	0.000
ANA 25F/25D	-	-	-	-	-	-	-	-	-	0.021
ANA 0F/100D	-	-	-	-	-	-	-	-	-	-

Displays resulting two-tailed p-values of Mann–Whitney U Tests.
Tests were conducted with a significance level of 5% (alpha = 0.05).
P-values above 0.05 represent tests in which the null hypothesis was not rejected (marked green).

Table 5.2: Series A - Mann–Whitney U Tests on Fitness - Part 2

	KNN					KNNA				
	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D
KNN 100F/0D	-	0.982	0.008	0.000	0.000	-	0.119	0.198	0.000	0.000
KNN 75F/25D	-	-	0.003	0.000	0.000	0.982	0.047	0.127	0.000	0.000
KNN 50F/50D	-	-	-	0.000	0.000	0.008	0.045	0.069	0.000	0.000
KNN 25F/25D	-	-	-	-	0.000	0.000	0.000	0.000	0.191	0.000
KNN 0F/100D	-	-	-	-	-	0.000	0.000	0.000	0.000	0.416
KNNA 100F/0D	-	-	-	-	-	-	0.119	0.198	0.000	0.000
KNNA 75F/25D	-	-	-	-	-	-	-	0.773	0.000	0.000
KNNA 50F/50D	-	-	-	-	-	-	-	-	0.000	0.000
KNNA 25F/25D	-	-	-	-	-	-	-	-	-	0.000
KNNA 0F/100D	-	-	-	-	-	-	-	-	-	-

Displays resulting two-tailed p-values of Mann–Whitney U Tests.
Tests were conducted with a significance level of 5% (alpha = 0.05).
P-values above 0.05 represent tests in which the null hypothesis was not rejected (marked green).

Table 5.3: Series A - Mann–Whitney U Tests on Fitness - Part 3

	KNN					KNNA				
	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D
AN 100F/0D	-	0.982	0.008	0.000	0.000	-	0.119	0.198	0.000	0.000
AN 75F/25D	0.416	0.524	0.015	0.000	0.000	0.416	0.355	0.424	0.000	0.000
AN 50F/50D	0.001	0.000	0.610	0.000	0.000	0.001	0.035	0.028	0.000	0.000
AN 25F/25D	0.000	0.000	0.000	0.000	0.183	0.000	0.000	0.000	0.000	0.625
AN 0F/100D	0.000	0.000	0.000	0.000	0.599	0.000	0.000	0.000	0.000	0.631
ANA 100F/0D	-	0.982	0.008	0.000	0.000	-	0.119	0.198	0.000	0.000
ANA 75F/25D	0.149	0.110	0.135	0.000	0.000	0.149	0.734	0.976	0.000	0.000
ANA 50F/50D	0.000	0.000	0.181	0.000	0.000	0.000	0.001	0.001	0.000	0.000
ANA 25F/25D	0.000	0.000	0.000	0.000	0.824	0.000	0.000	0.000	0.000	0.451
ANA 0F/100D	0.000	0.000	0.000	0.000	0.011	0.000	0.000	0.000	0.000	0.076

Displays resulting two-tailed p-values of Mann–Whitney U Tests.
Tests were conducted with a significance level of 5% (alpha = 0.05).
P-values above 0.05 represent tests in which the null hypothesis was not rejected (marked green).

Table 5.4: Series A - Mann–Whitney U Tests on Diversity - Part 1

	AN					ANA				
	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D
AN 100F/0D	-	0.000	0.000	0.000	0.000	0.061	0.000	0.000	0.000	0.000
AN 75F/25D	-	-	0.000	0.000	0.000	0.000	0.853	0.000	0.000	0.000
AN 50F/50D	-	-	-	0.000	0.000	0.000	0.000	0.133	0.000	0.001
AN 25F/25D	-	-	-	-	0.109	0.000	0.000	0.000	0.412	0.663
AN 0F/100D	-	-	-	-	-	0.000	0.000	0.000	0.900	0.947
ANA 100F/0D	-	-	-	-	-	-	0.000	0.000	0.000	0.000
ANA 75F/25D	-	-	-	-	-	-	-	0.000	0.000	0.000
ANA 50F/50D	-	-	-	-	-	-	-	-	0.000	0.000
ANA 25F/25D	-	-	-	-	-	-	-	-	-	0.830
ANA 0F/100D	-	-	-	-	-	-	-	-	-	-

Displays resulting two-tailed p-values of Mann–Whitney U Tests.
Tests were conducted with a significance level of 5% (alpha = 0.05).
P-values above 0.05 represent tests in which the null hypothesis was not rejected (marked green).

Table 5.5: Series A - Mann–Whitney U Tests on Diversity - Part 2

	KNN					KNNA				
	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D
KNN 100F/0D	-	0.000	0.000	0.000	0.000	0.762	0.000	0.000	0.000	0.000
KNN 75F/25D	-	-	0.000	0.000	0.000	0.000	0.277	0.003	0.000	0.000
KNN 50F/50D	-	-	-	0.000	0.000	0.000	0.000	0.228	0.000	0.000
KNN 25F/25D	-	-	-	-	0.102	0.000	0.000	0.000	0.982	0.023
KNN 0F/100D	-	-	-	-	-	0.000	0.000	0.000	0.099	0.483
KNNA 100F/0D	-	-	-	-	-	-	0.000	0.000	0.000	0.000
KNNA 75F/25D	-	-	-	-	-	-	-	0.000	0.000	0.000
KNNA 50F/50D	-	-	-	-	-	-	-	-	0.000	0.000
KNNA 25F/25D	-	-	-	-	-	-	-	-	-	0.014
KNNA 0F/100D	-	-	-	-	-	-	-	-	-	-

Displays resulting two-tailed p-values of Mann–Whitney U Tests.
Tests were conducted with a significance level of 5% (alpha = 0.05).
P-values above 0.05 represent tests in which the null hypothesis was not rejected (marked green).

Table 5.6: Series A - Mann–Whitney U Tests on Diversity - Part 3

	KNN					KNNA				
	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D
AN 100F/0D	0.000	0.000	0.000	0.000	0.000	0.039	0.000	0.000	0.000	0.000
AN 75F/25D	0.000	0.115	0.000	0.000	0.000	0.000	0.355	0.000	0.000	0.000
AN 50F/50D	0.000	0.000	0.196	0.080	0.004	0.000	0.000	0.042	0.080	0.001
AN 25F/25D	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
AN 0F/100D	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
ANA 100F/0D	0.569	0.000	0.000	0.000	0.000	0.985	0.000	0.000	0.000	0.000
ANA 75F/25D	0.000	0.077	0.000	0.000	0.000	0.000	0.271	0.000	0.000	0.000
ANA 50F/50D	0.000	0.000	0.773	0.000	0.000	0.000	0.000	0.348	0.000	0.000
ANA 25F/25D	0.000	0.000	0.000	0.007	0.045	0.000	0.000	0.000	0.009	0.064
ANA 0F/100D	0.000	0.000	0.000	0.012	0.077	0.000	0.000	0.000	0.015	0.105

Displays resulting two-tailed p-values of Mann–Whitney U Tests.
Tests were conducted with a significance level of 5% (alpha = 0.05).
P-values above 0.05 represent tests in which the null hypothesis was not rejected (marked green).

Table 5.7: Series A - Combined Mann–Whitney U Tests - Part 1

	AN					ANA				
	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D
AN 100F/0D	-	↑	×	×	×	×	↑	×	×	×
AN 75F/25D	-	-	×	×	×	←	×	×	×	×
AN 50F/50D	-	-	-	×	×	×	×	×	×	×
AN 25F/25D	-	-	-	-	×	×	×	×	×	×
AN 0F/100D	-	-	-	-	-	×	×	×	×	×
ANA 100F/0D	-	-	-	-	-	-	↑	×	×	×
ANA 75F/25D	-	-	-	-	-	-	-	×	×	×
ANA 50F/50D	-	-	-	-	-	-	-	-	×	×
ANA 25F/25D	-	-	-	-	-	-	-	-	-	×
ANA 0F/100D	-	-	-	-	-	-	-	-	-	-

Displays which experiment comparisons support the goal of this research (marked green).
*Green cells represent the **success** to reject the null hypothesis during **fitness** tests (Fig 5.1).*
*Green cells represent the **failure** to reject the null hypothesis during **diversity** tests (Fig 5.4).*
Arrows indicate which experiment from the comparison achieved the higher diversity score.

Table 5.8: Series A - Combined Mann–Whitney U Tests - Part 2

	KNN					KNNA				
	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D
KNN 100F/0D	-	↑	×	×	×	×	↑	↑	×	×
KNN 75F/25D	-	-	×	×	×	↑	×	↑	×	×
KNN 50F/50D	-	-	-	×	×	×	×	×	×	×
KNN 25F/25D	-	-	-	-	×	×	×	×	×	×
KNN 0F/100D	-	-	-	-	-	×	×	×	×	×
KNNA 100F/0D	-	-	-	-	-	-	↑	↑	×	×
KNNA 75F/25D	-	-	-	-	-	-	-	↑	×	×
KNNA 50F/50D	-	-	-	-	-	-	-	-	×	×
KNNA 25F/25D	-	-	-	-	-	-	-	-	-	×
KNNA 0F/100D	-	-	-	-	-	-	-	-	-	-

*Displays which experiment comparisons support the goal of this research (marked green). Green cells represent the **success** to reject the null hypothesis during **fitness** tests (Fig 5.2). Green cells represent the **failure** to reject the null hypothesis during **diversity** tests (Fig 5.5). Arrows indicate which experiment from the comparison achieved the higher diversity score.*

Table 5.9: Series A - Combined Mann–Whitney U Tests - Part 3

	KNN					KNNA				
	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D
AN 100F/0D	↑	↑	×	×	×	↑	↑	↑	×	×
AN 75F/25D	←	×	×	×	×	↑	×	↑	×	×
AN 50F/50D	×	×	×	×	×	×	×	×	×	×
AN 25F/25D	×	×	×	×	←	×	×	×	×	←
AN 0F/100D	×	×	×	×	←	×	×	×	×	←
ANA 100F/0D	×	↑	×	×	×	×	↑	↑	×	×
ANA 75F/25D	←	×	↑	×	×	↑	×	↑	×	×
ANA 50F/50D	×	×	×	×	×	×	×	×	×	×
ANA 25F/25D	×	×	×	×	←	×	×	×	×	×
ANA 0F/100D	×	×	×	×	×	×	×	×	×	×

*Displays which experiment comparisons support the goal of this research (marked green). Green cells represent the **success** to reject the null hypothesis during **fitness** tests (Fig 5.2). Green cells represent the **failure** to reject the null hypothesis during **diversity** tests (Fig 5.5). Arrows indicate which experiment from the comparison achieved the higher diversity score.*

A total of 32 Mann–Whitney U tests provide results that appear to achieve the goal of this research, but this includes comparisons between diversity-based evaluation strategies. Looking exclusively at tests comparing a diversity-based evaluation strategy with the fitness-based baseline it corresponds to, a total of five experiments truly achieve the goal of this research:

1. AN 75F/25D
2. ANA 75F/25D
3. KNN 75F/25D
4. KNNA 75F/25D
5. KNNA 50F/50D

Throughout the empirical analyses of the weightings, the qualitative behaviours of 75F/25D weightings were unsatisfactory. The level of diversity was simply too small for enough interesting behaviours to emerge. Some could be seen, but not enough to warrant a full qualitative behaviour analysis, hence their omission. Fortunately, KNNA 50F/50D is included in the list of experiments that statistically achieve the goal of this research. From this, and the successful quantitative and qualitative behaviour analyses conducted for KNNA 50F/50D, it can be concluded that KNNA 50F/50D is a successful strategy for evolving diverse and effective behaviours in pursuit domains.

Chapter 6

Experiment Series B: Predator Versus Advanced Prey

Experiment Series B is the second set of experiments and is primarily used as a supplementary series to re-examine the results from Experiment Series A. From the results in Experiment Series A, it was determined that KNNA is the best of the four strategies at achieving the goal of this research. As such, AN, ANA, and KNN will no longer be explored. Instead, the KNNA experiments will be repeated with a variation applied to the domain: advanced prey.

6.1 Baseline

The goal for this baseline was to provide a variation to the domain that removes the excessive use of the scraping behaviour. As discussed in section 3.2.2, two significant upgrades were added to the prey: bouncing and fleeing.

As seen in Figure 6.1, the population trend already shows a decrease in the fitness score compared to the baseline in Experiment Series A. It makes sense considering how simple the prey used to be; the GP has to evolve much more complex behaviours to catch these new prey.

The qualitative analysis brings insight to the new technique used to catch the prey, and unfortunately, its not very interesting at all. Figure 6.2 shows that the

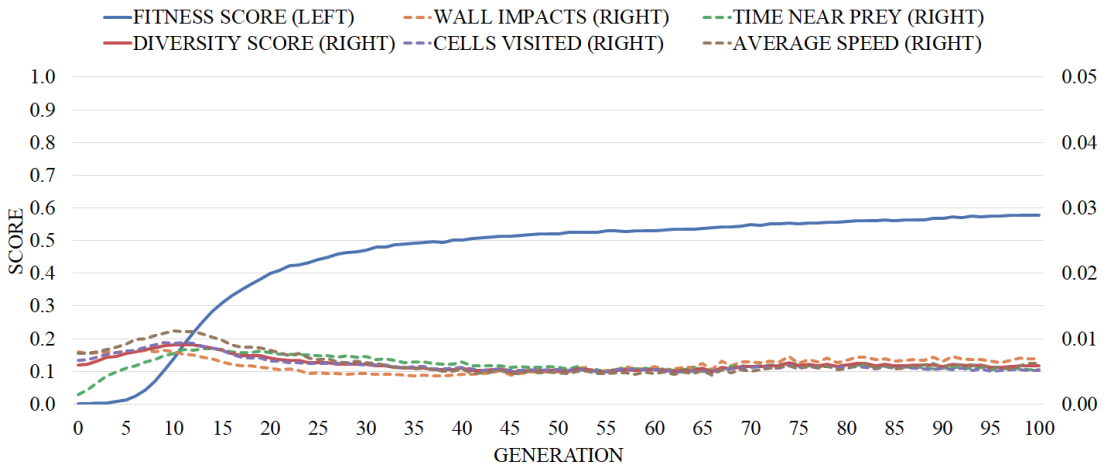


Figure 6.1: Series B - Baseline 100F/0D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

new behaviour that the GP has evolved to exploit involves simply waiting for a prey to approach before pouncing. This is analogous to ambush/sit-and-wait predators found throughout nature, barring the motivation to conserve energy. Occasionally, the predator would scan the environment, or even travel in a circular motion, both of which allow more cells to be explored. Despite this, sitting idly appears to be more valuable to to the GP throughout KNNA 100F/0D.

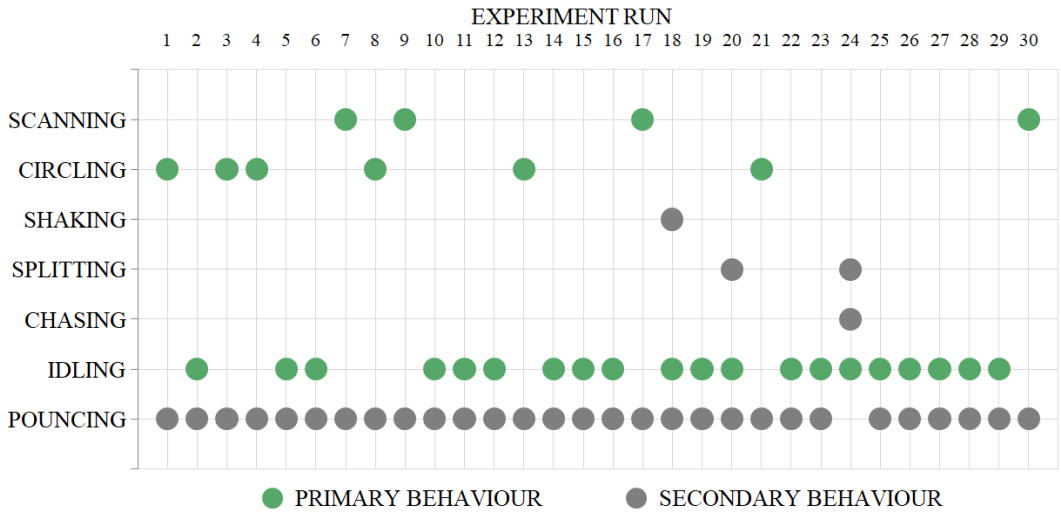


Figure 6.2: Series B - Baseline 100F/0D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation.

6.2 K-Nearest Neighbours with Archive

Applying the KNNA strategy in this domain immediately showed promising results. As shown in Figure 6.3, a local optima breakout occurs around generation 70. Despite the diversity score suddenly increasing, the fitness score maintains itself around the 50% it converged to around generation 50. Once again, the 0F/100D weighting resulted in a terrible fitness score (Figure 6.4). The diversity score continues to increase with some curious instabilities.

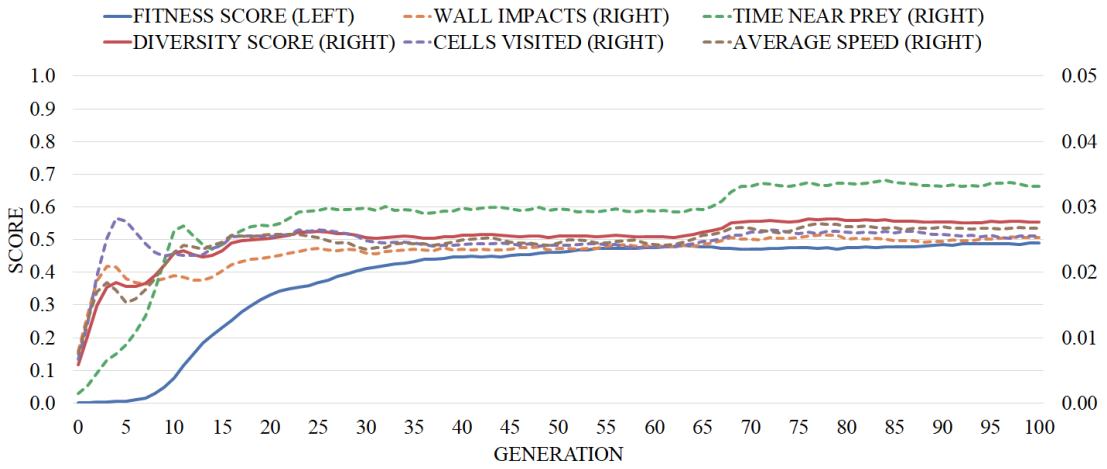


Figure 6.3: Series B - KNNA 50F/50D - Population Trend
Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

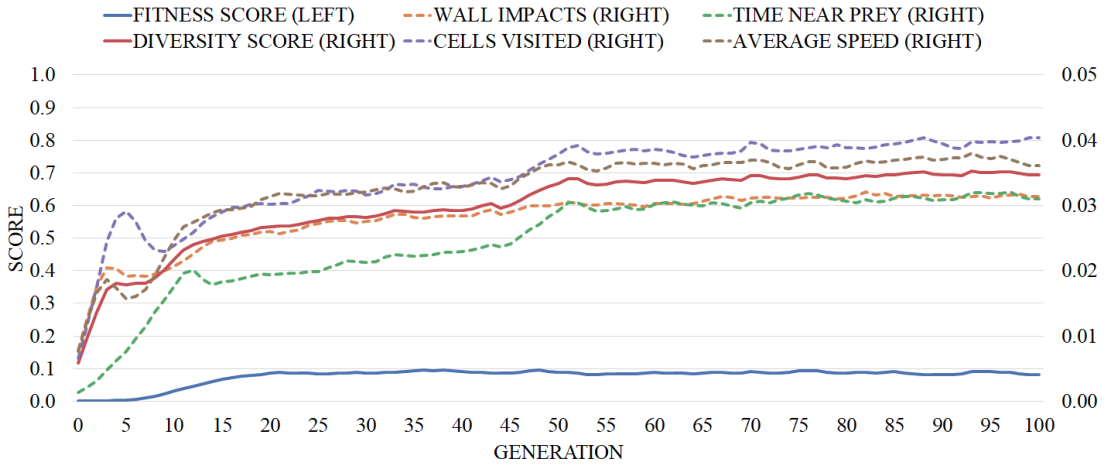


Figure 6.4: Series B - KNNA 0F/100D - Population Trend
Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

The final scores shown in Figure 6.5 follow a similar pattern to previous KNNA experiment sets. KNNA 50F/50D (the best of Experiment Series A) did a phenomenal job once again. While the fitness score only drops by 4%, the diversity score increases by 15% (almost quadrupling). The statistical significance of this will be tested like before during the discussion portion of this chapter.

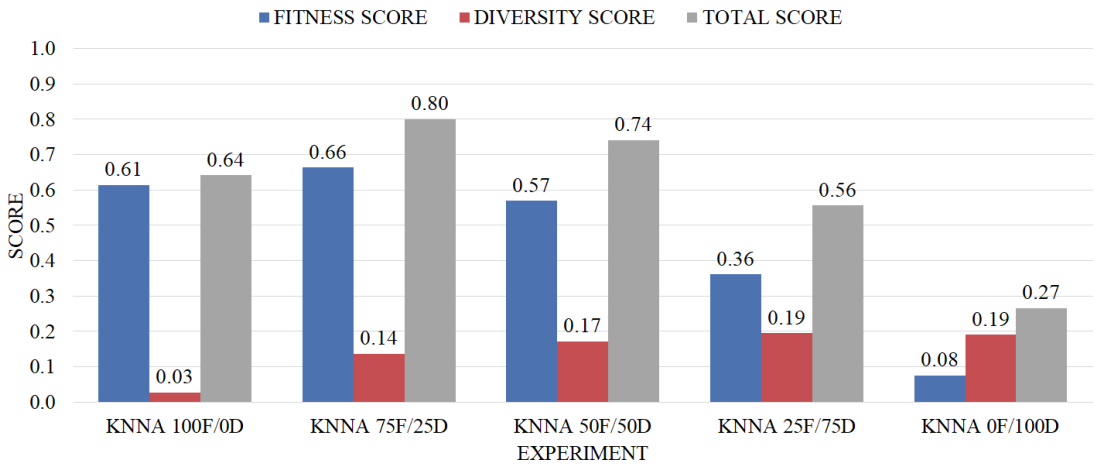


Figure 6.5: Series B - KNNA - Final Scores
Average scores of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations. Total score is supplementary and was not used by the GP.

Moving on to the qualitative analysis, we can see an incredibly congested collection of behaviour tallies for the KNNA 50F/50D experiment (Figure 6.6). Now that prey bounce off of walls, we no longer see the typical scraping behaviour being overused. Instead, a healthy mix of idling and wandering are used as primary behaviours, with a little bit of scanning and circling as well.

Just like in Experiment Series A, the simulation replays of KNNA 50F/50D in this series is packed with interesting behaviours. Many simulations involve the predator behaving in sporadic and complex ways. It becomes difficult to describe many of these emergent behaviours, but they are interesting nonetheless. For example, sometimes the predator will move in small circles with a slow speed, and when a prey is sensed, the predator will begin shaking and splitting in the direction of the prey. Sometimes the predator will scan across the environment in a sine-wave fashion whilst splitting the field of view in two opposite directions,

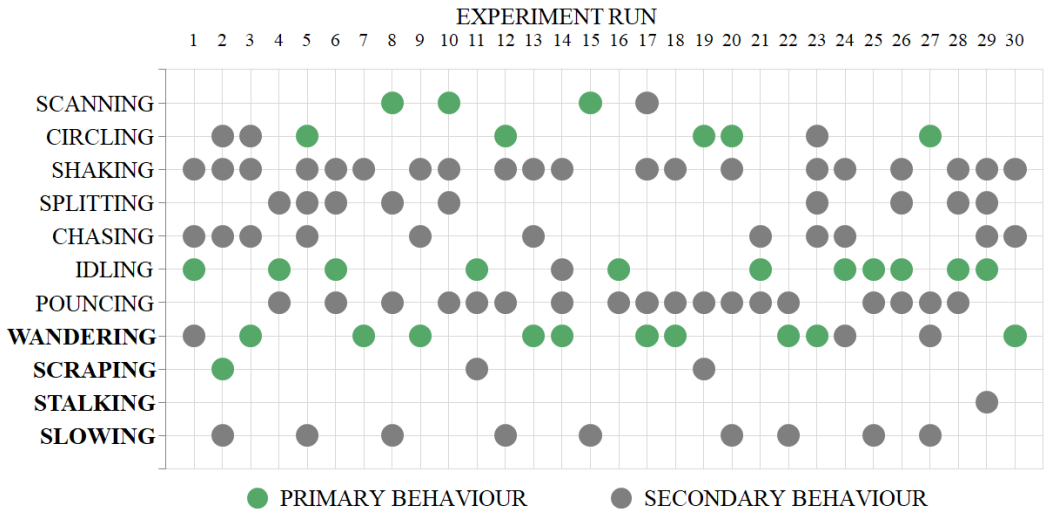


Figure 6.6: Series B - KNNA 50F/50D - Qualitative Behaviours
Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

creating an hourglass-esque perception area. There are many other interesting combinations of behaviours emerging throughout this experiment (far too many to mention). The primary importance to note is that KNNA 50F/50D succeeded once again.

6.3 Series B Discussion

Mann–Whitney U tests were conducted on the strategies used through Experiment Series B. The interpretations explained in Experiment Series A (section 5.6) were applied to this experiment series as well. The majority of tests are supplementary, attention should be paid primarily to KNNA 50F/50D.

Once again, KNNA 50F/50D achieved the goal of this research statistically. The Mann–Whitney U test between KNNA 100F/0D (the baseline) and KNNA 50F/50D shows no statistically significant difference between the fitness scores. The Mann–Whitney U test between these two experiments also shows that there *is* a statistically significant difference between the *diversity* scores.

Table 6.1: Series B - Mann–Whitney U Tests on Fitness

	KNNA				
	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D
KNNA 100F/0D	-	0.029	0.351	0.000	0.000
KNNA 75F/25D	-	-	0.003	0.000	0.000
KNNA 50F/50D	-	-	-	0.000	0.000
KNNA 25F/25D	-	-	-	-	0.000
KNNA 0F/100D	-	-	-	-	-

Displays resulting two-tailed p-values of Mann–Whitney U Tests.
Tests were conducted with a significance level of 5% (alpha = 0.05).
P-values above 0.05 represent tests in which the null hypothesis was not rejected (marked green).

Table 6.2: Series B - Mann–Whitney U Tests on Diversity

	KNNA				
	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D
KNNA 100F/0D	-	0.000	0.000	0.000	0.000
KNNA 75F/25D	-	-	0.000	0.000	0.000
KNNA 50F/50D	-	-	-	0.003	0.000
KNNA 25F/25D	-	-	-	-	0.000
KNNA 0F/100D	-	-	-	-	-

Displays resulting two-tailed p-values of Mann–Whitney U Tests.
Tests were conducted with a significance level of 5% (alpha = 0.05).
P-values above 0.05 represent tests in which the null hypothesis was not rejected (marked green).

Combining the Mann–Whitney U tests with the increase in diversity seen in Figure 6.5 and the incredibly interesting qualitative behaviours seen in Figure 6.6, KNNA 50F/50D is once again proven (quantitatively, qualitatively, and statistically) to achieve the goal of this research.

Chapter 7

Experiment Series C: Food Gathering

Experiment Series C is the final experiment series and, like Experiment Series B, it used as a supplementary series to further examine the results from Experiment Series A. This series includes an extreme change to the way prey behave; their speed is reduced to zero. This essentially transforms Predator Versus Prey into a Food Gathering simulation, a domain in which an agent must forage for food throughout its environment. Besides the speed reduction, no other changes were made. There are still twenty "prey" agents and the means of catching them are the same as before.

7.1 Baseline

The baseline analysis begins with the 100F/0D population trend (Figure 7.1). Unlike in the previous experiment series, the predators could not even manage to catch half of the prey. The fitness and diversity trends are both disappointingly low. A low diversity score was expected, but it was assumed that the fitness score would have been the highest of the three experiment series.

Intriguingly, the lack of movement on the preys' behalf makes the task of catching them *more* difficult for the predator. The system and language were designed

specifically for literal "pursuit" scenarios (agents chasing agents). There must be an oversight somewhere within the language that prevents the predator from achieving a score as high as it could when the prey had movement.

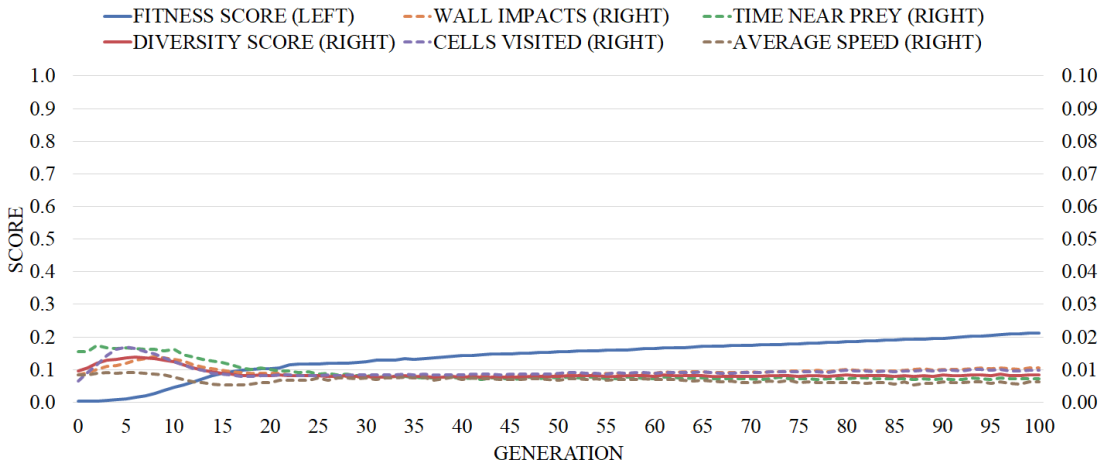


Figure 7.1: Series C - 100F/0D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

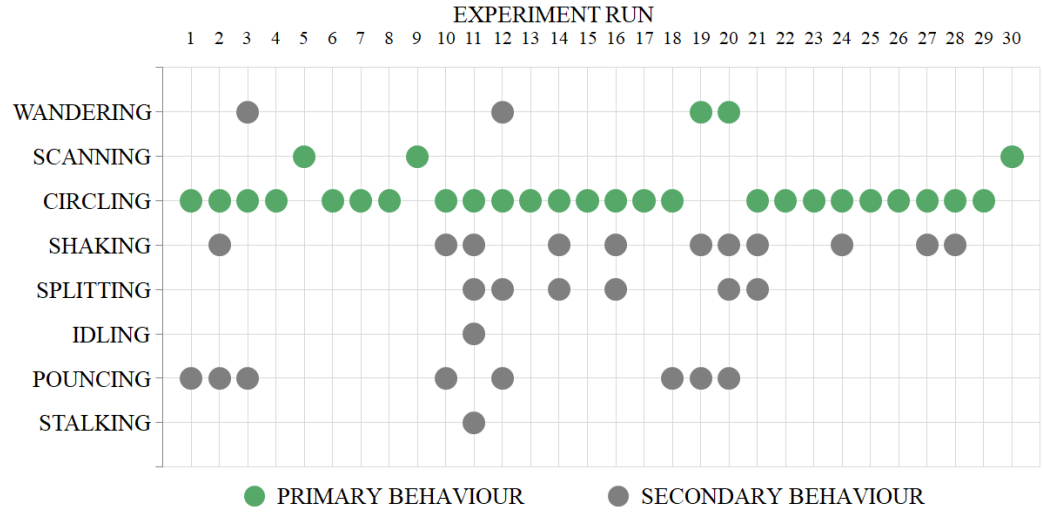


Figure 7.2: Series C - 100F/0D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation.

7.2 K-Nearest Neighbours with Archive

Similarly to previous experiments, KNNA is managing to break out of local optima. Figure 7.3 shows the GP escape a local optima *four times* before reaching the generation limit. It can be assumed that this would have continued further, and provides more motivation for future work with extended generation limits.

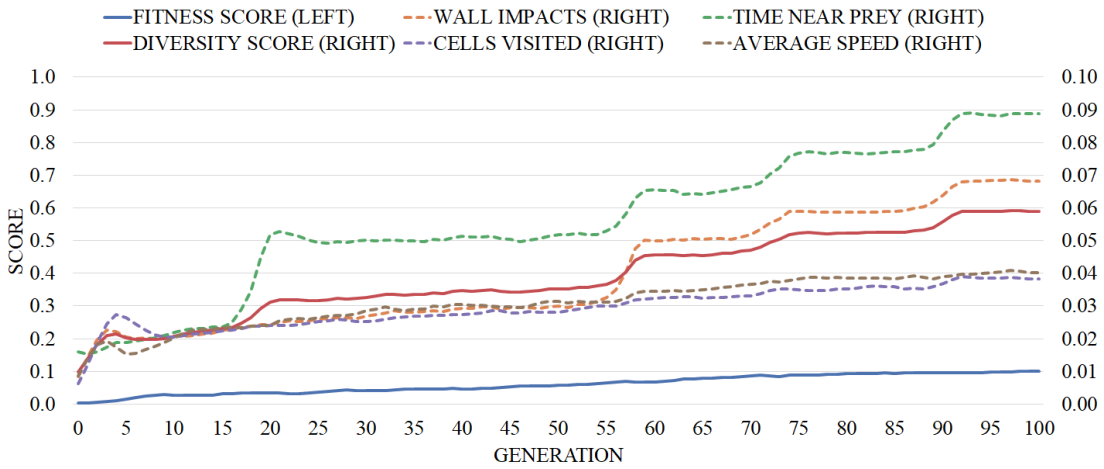


Figure 7.3: Series C - KNNA 50F/50D - Population Trend
Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

The most influential quantitative behaviour in this population trend is time near prey, which makes sense. Now that the prey cannot move, the predator is in full control of how long it spends near them. With this new strength, the predator is making great strides with the diversity score.

The final scores are quite disappointing for this experiment set. The highest fitness score (35%) belongs to the 75F/25D weighting and immediately suffers by 19% following that. KNNA 0F/100D appears to hold the lowest fitness score achieved in this research: 2%. This can be seen in Figure 7.4.

Lastly, a look into the qualitative analysis of KNNA 50F/50D shows another successful set of interesting behaviours. The many behaviours emerging throughout this experiment keeps the simulation interesting and entertaining, despite the prey not even moving. A comedic note to make is the existence of the stalking behaviour. Despite the prey not moving, the predator would sometimes stalk them.

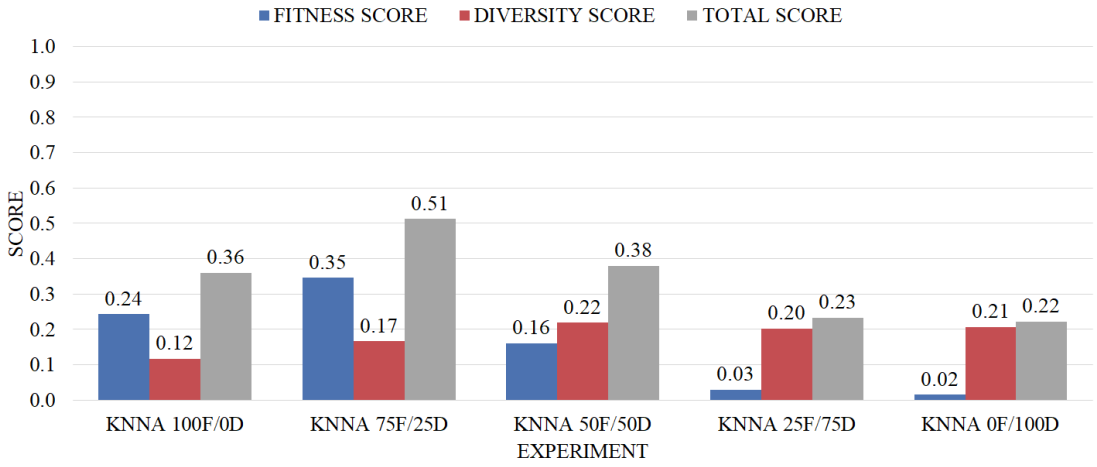


Figure 7.4: Series C - KNNA - Final Scores
Average scores of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations. Total score is supplementary and was not used by the GP.

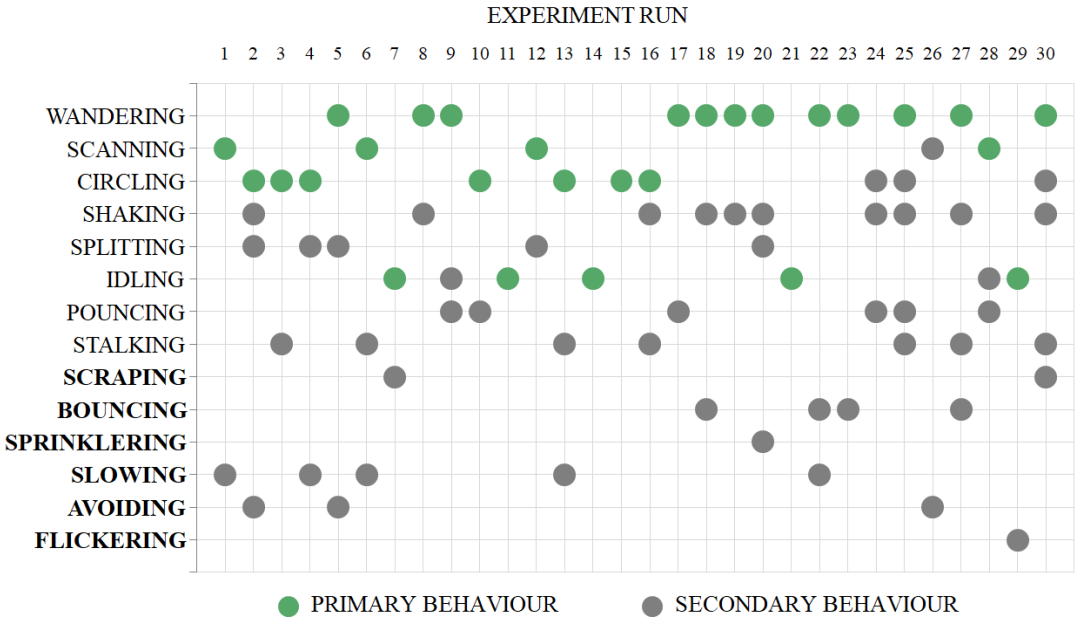


Figure 7.5: Series C - KNNA 50F/50D - Qualitative Behaviours
Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

7.3 Series C Discussion

Ultimately, Experiment Series C was successful at providing an interesting set of emergent behaviours, but the drop in the fitness score was too much. Performing one last set of Mann–Whitney U tests confirms this, where Figure 7.1 shows the rejection of the null hypothesis between the fitness scores of KNNA 100F/0D and KNNA 50F/0D.

Table 7.1: Series C - Mann–Whitney U Tests on Fitness

	KNNA				
	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D
KNNA 100F/0D	-	0.021	0.018	0.000	0.000
KNNA 75F/25D	-	-	0.000	0.000	0.000
KNNA 50F/50D	-	-	-	0.000	0.000
KNNA 25F/25D	-	-	-	-	0.751
KNNA 0F/100D	-	-	-	-	-

Displays resulting two-tailed p-values of Mann–Whitney U Tests.
Tests were conducted with a significance level of 5% (alpha = 0.05).
P-values above 0.05 represent tests in which the null hypothesis was not rejected (marked green).

Table 7.2: Series C - Mann–Whitney U Tests on Diversity

	KNNA				
	100F 0D	75F 25D	50F 50D	25F 75D	0F 100D
KNNA 100F/0D	-	0.000	0.000	0.000	0.000
KNNA 75F/25D	-	-	0.000	0.000	0.000
KNNA 50F/50D	-	-	-	0.206	0.109
KNNA 25F/25D	-	-	-	-	0.865
KNNA 0F/100D	-	-	-	-	-

Displays resulting two-tailed p-values of Mann–Whitney U Tests.
Tests were conducted with a significance level of 5% (alpha = 0.05).
P-values above 0.05 represent tests in which the null hypothesis was not rejected (marked green).

Chapter 8

Conclusion

This research explored strategies designed to encourage diverse behaviours while still maintaining an appreciable level of efficacy. The nature of fitness-based evaluation tends to exploit particular behaviours in monotonous and uninteresting ways, but with the addition of diversity-based evaluation, this is remedied.

K-Nearest Neighbours with Archive (KNNA) with an even 50/50 split between the fitness score and the diversity score (50F/50D) achieved the best results throughout this research. KNNA 50F/50D proved itself quantitatively and qualitatively in more than one domain. Other strategies provided excellent balances between fitness and diversity, and some strategies even *increased* fitness, but KNNA 50F/50D was the most successful at achieving the goals of this research.

Experiment Series A thoroughly examined KNNA 50F/50D and revealed its qualitative, quantitative, and statistical success at achieving the goal of this research. Experiment Series B provided all the same accomplishments for KNNA 50F/50D as Experiment Series A did, even though the task's difficulty was significantly increased. Unfortunately, Experiment Series C could not support KNNA 50F/50D statistically, but the quantitative and qualitative analysis still showed appreciable results. All figures relating to these experiments can be seen throughout the appendices, organized by type of data.

It should be remembered that the systems for this research were designed for literal pursuit, not food gathering. Therefore, a system more suited toward food

gathering (perhaps a different GP language) should be created to test the affects of KNNA 50F/50D (instead of just reducing the prey speed to zero).

To reiterate, this research was a success. A strategy was found with which diverse behaviours could be observed quantitatively and qualitatively. This strategy was thoroughly examined and showed no statistically significant damage to the fitness score. The behaviours were effective, diverse, and most importantly, interesting.

8.1 Comparable Work

8.1.1 K-Nearest Neighbours

Similar work [27][38][30] would use K-Nearest Neighbours when evaluating the diversity for an individual. They would also utilize an archive of previous solutions spanning across multiple previous generations. This was the inspiration for KNNA: the most successful strategy used throughout this research. It was assumed that KNNA would be the most successful strategy due to its popular usage throughout related work, but the other strategies (AN, ANA, and KNN) were explored anyways for the sake of completion.

8.1.2 Multi-objectivization

As mentioned above, similar work [27] would use K-Nearest Neighbours when evaluating for diversity, but the technique used to measure the diversity among multiple behaviours was handled differently for this research. While other work would use the Euclidean distances between the characterization vectors (with the characterization vector components reflecting each behaviour) [49], this research used sum of ranks to achieve a score for the distances of each separate component in the characterization vectors, as mentioned in Chapter 3. The differences between these techniques were not the focus of this research, but empirical analysis suggests that the use of sum of ranks mitigates the risk of particular behaviours dominating

the score.

8.1.3 Emergent Behaviours

While other work [50] [7] has proven that the emergence of high-level behaviours are possible, the emergence of *interesting* high-level behaviours is a neglected topic. This research expanded on the search for emergent behaviours by finding strategies to encourage a *diverse set* of emergent behaviours. Applications like video-games (specifically predator versus prey) can greatly benefit from this type of research. Work from Dockhorn and Kruse [18] evolved multiple opponents separately to give the appearance of diversity, but this paper successfully increased diversity among a *single* agent. This could then be applied to the multiple opponents (in Dockhorn and Kruse's example) to greatly enhance the degree of diversity. Additionally, other work (like Liberatore *et. al.*) explored strategies that utilized real-time evolution. The evolution in this paper was not real-time, but was successful at producing a diverse set of solutions that could be easily swapped for one another during a real-time application. This entirely avoids the compromises that must be made when budgeting the devices resources.

8.1.4 Continuous Environment

It seems common for work in this domain [7] to use discrete environments, limiting the agents to the four cardinal directions. In some work, this is slightly upgraded by allowing the agents to move diagonally, but this research ensured that the agents had the ability to move in any direction. The intention is to push this type of research further in the direction of modern video-game applications, as most modern video-games are being developed with continuous environments. Additionally, while it was not the intention to compare continuous environments with discrete environments, empirical analyses suggested that continuous environments improved the overall complexity of behaviours (rightfully so, as the behaviour search space is much larger).

8.2 Future Work

Many features were designed and used during the development stages of this research, some of which were removed for a variety of reasons. The list of removed features is briefly described below, and shall be included in future research:

1. **Obstacles**: The environment was initially designed to handle the placement of wall cells *anywhere*. Rooms and corridors could be built for the agents to navigate through. This was removed to simplify the predator's perception of the environment.

2. **Toroidal Environment**: The environment was upgraded to have looping edges. If an agent were to reach the edge of the environment, they would loop back to the opposite edge. This environment's geometry is best described as a toroid. This was removed in favour of a more "realistic" environment.

3. **Trials**: The evaluation of solutions requires the GP to control a predator throughout a simulation. This simulation contained some randomized components (start locations and prey movement) so "luck" had a small factor to play into the final score. Adding trials was a means to mitigate this by evaluating a solution over multiple simulations and averaging the score. This was removed after discovering it provided no benefits, since the behaviour of the GP's trees were generally stable.

4. **Predator Starvation**: The predator was given a starvation mechanic to encourage catching prey. If the predator failed to catch a prey within a specific amount of time units, the predator would die and the simulation would end. This was removed due to the behaviour search space being narrowed by this unnecessary requirement.

5. **Time-based Scoring**: Points were scaled based on the time they were acquired. Predators were rewarded with points that became less valuable as time went on. For example, a prey caught near the beginning of the

simulation would be worth near a full point, while a prey caught near the end of the simulation would be worth only a small portion of a point. This was removed to simplify the interpretation of fitness scores.

There were also many features and concepts that were considered but not developed throughout this research. These features and concepts appear to be worth exploring but were out of the scope at this time of writing. Future development of this research's systems will likely see the inclusion of:

1. **Real-time Evolution**: Modern video-games utilize graphics processing units (GPUs) and it has been shown that GPs can make use of the many threads provided by those GPUs [51] [52]. While some works [53] have dismissed the idea of evolving in real-time due to the computational resources required to do so, others have welcomed the idea of moving evolution to real-time [17] [54] [55]. This would make way for *significantly* more interesting opponents, especially considering how many modern video-games (*Metal Gear Solid V: The Phantom Pain* [56] and *Escape from Tarkov* [57] for example) spend time developing opponents that mimic real-time adaption to the players.

2. **MAP-Elites**. MAP-Elites is designed to handle searches in high-dimensional space, and since these experiments can use many behaviour measures, MAP-Elites might be helpful. This has been discussed and explored in previous works and shows promise [58] [59].

3. **Surprise Search**. Surprise search is another strategy to encourage diversity when evaluating solutions. While novelty search discourages repetitions of old solutions, surprise search discourages predicted *future* solutions. Surprise has been studied with success by Gravina *et. al* and shows great promise [60] [61] [62].

4. **Coevolution**. Coevolution can be implemented as competitive [37] or co-operative [38]. As competitive coevolution, the prey would be given the ability to evolve as well. This will lead to an arms-race between the predator

and prey, hopefully encouraging a more steady evolution. As cooperative evolution, multiple predators would be added to the arena and simple communication between predators would be added to the language. This would hopefully encourage some interesting tactics between groups of agents.

5. **Other Applications**. There are many other applications, like herding, that could benefit from diverse agents. Herding is another application of embodied agents akin to a dog grouping sheep. Herding was an emergent behaviour observed during this research, so redesigning the system to task the GP with a herding problem (like those explored by Licitra *et. al.* [63]) may yield worthwhile results.

Bibliography

[1] B. Bennett and M. Trafankowski, "A comparative investigation of herding algorithms," in *Proc. Symp. on Understanding and Modelling Collective Phenomena (UMoCoP)*, pp. 33–38, 2012.

[2] J. Brulé, K. Engel, N. Fung, and I. Julien, "Evolving shepherding behavior with genetic programming algorithms," *arXiv preprint arXiv:1603.06141*, 2016.

[3] J. R. Koza, J. Roughgarden, and J. P. Rice, "Evolution of food-foraging strategies for the caribbean anolis lizard using genetic programming.," *Adapt. Behav.*, vol. 1, no. 2, pp. 171–199, 1992.

[4] M. Sinclair and S. Shami, "Evolving simple software agents: comparing genetic algorithm and genetic programming performance," in *Second International Conference On Genetic Algorithms In Engineering Systems: Innovations And Applications*, pp. 421–426, IET, 1997.

[5] T. Haynes and S. Sen, "Evolving behavioral strategies in predators and prey," in *International Joint Conference on Artificial Intelligence*, pp. 113–126, Springer, 1995.

[6] M. Laumanns, G. Rudolph, and H.-P. Schwefel, "A spatial predator-prey approach to multi-objective optimization: A preliminary study," in *International Conference on Parallel Problem Solving from Nature*, pp. 241–249, Springer, 1998.

[7] G. Grossi and B. Ross, "Evolved communication strategies and emergent behaviour of multi-agents in pursuit domains," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 110–117, IEEE, 2017.

[8] S. Luke and L. Spector, "Evolving teamwork and coordination with genetic programming," *Genetic Programming*, vol. 96, 1996.

[9] S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler, "Co-evolving soccer softbot team coordination with genetic programming," in *Robot Soccer World Cup*, pp. 398–411, Springer, 1997.

[10] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-agent Systems*, vol. 11, no. 3, pp. 387–434, 2005.

[11] S. Luke, "Assisted parameter and behavior calibration in agent-based models with distributed optimization," in *Advances in Practical Applications of Agents, Multi-Agent Systems, and Trustworthiness. The PAAMS Collection: 18th International Conference, PAAMS 2020, L'Aquila, Italy, October 7-9, 2020, Proceedings*, vol. 12092, p. 93, Springer Nature, 2020.

[12] Y. Demazeau, T. Holvoet, J. M. Corchado, and S. Costantini, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Trustworthiness. The PAAMS Collection: 18th International Conference, PAAMS 2020, L'Aquila, Italy, October 7-9, 2020, Proceedings*, vol. 12092. Springer Nature, 2020.

[13] T. Haynes, S. Sen, D. Schoenefeld, and R. Wainwright, "Evolving multiagent coordination strategies with genetic programming," in *Technical Report UTULSA-MCS-95–04*, 1995.

[14] D. G. Wilson, S. Cussat-Blanc, H. Luga, and J. F. Miller, "Evolving simple programs for playing atari games," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 229–236, 2018.

[15] T. Bullen and M. Katchabaw, "Using genetic algorithms to evolve character behaviours in modern video games," *Proceedings of the GAMEON-NA*, 2008.

[16] General Computer Corporation and Midway, *Ms. Pac-Man*. Midway, 1982.

[17] F. Liberatore, A. M. Mora, P. A. Castillo, and J. J. Merelo, "Comparing heterogeneous and homogeneous flocking strategies for the ghost team in the game of ms. pac-man," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 3, pp. 278–287, 2015.

[18] A. Dockhorn and R. Kruse, "Combining cooperative and adversarial coevolution in the context of pac-man," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 60–67, IEEE, 2017.

[19] A. M. Alhejali and S. M. Lucas, "Evolving diverse ms. pac-man playing agents using genetic programming," in *2010 UK Workshop on Computational Intelligence (UKCI)*, pp. 1–6, IEEE, 2010.

[20] P. Rohlfshagen and S. M. Lucas, "Ms pac-man versus ghost team cec 2011 competition," in *2011 IEEE Congress of Evolutionary Computation (CEC)*, pp. 70–77, IEEE, 2011.

[21] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local competition," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, pp. 211–218, 2011.

[22] H. Aslam and J. A. Brown, "Affordance theory in game design: a guide toward understanding players," *Synthesis Lectures on Games and Computational Intelligence*, vol. 4, no. 1, pp. 1–111, 2020.

[23] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary Computation*, vol. 19, no. 2, pp. 189–223, 2011.

[24] J. Lehman and K. O. Stanley, "Novelty search and the problem with objectives," in *Genetic Programming Theory and Practice IX*, pp. 37–56, Springer, 2011.

[25] J.-B. Mouret and S. Doncieux, "Encouraging behavioral diversity in evolutionary robotics: An empirical study," *Evolutionary Computation*, vol. 20, no. 1, pp. 91–133, 2012.

[26] A. Team, "Automated game testing through novelty search and quality diversity, application to space engineers," *Internship Proposal, Sorbonne University*, 2020.

[27] J.-B. Mouret, "Novelty-based multiobjectivization," in *New horizons in Evolutionary Robotics*, pp. 139–154, Springer, 2011.

[28] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," Kluwer Academic Publishers-Plenum Publishers; Kluwer Academic Publishers, 1988.

[29] J. Lehman, K. O. Stanley, *et al.*, "Exploiting open-endedness to solve problems through the search for novelty," in *Proceedings of the 11th International Conference on Artificial Life (ALIFE XI)*, pp. 329–336, 2008.

[30] J. Gomes, P. Mariano, and A. L. Christensen, "Devising effective novelty search algorithms: A comprehensive empirical study," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 943–950, 2015.

[31] J. Lehman, K. O. Stanley, and R. Miikkulainen, "Effective diversity maintenance in deceptive domains," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pp. 215–222, 2013.

[32] Creative Assembly, *Alien: Isolation*. Sega, 2014.

[33] J. Gomes, P. Urbano, and A. L. Christensen, "Introducing novelty search in evolutionary swarm robotics," in *International Conference on Swarm Intelligence*, pp. 85–96, Springer, 2012.

[34] J. R. Koza and J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.

[35] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. Lulu Press, 2008.

[36] G. N. Yannakakis and A. Liapis, "Searching for surprise," in *Proceedings of the International Conference on Computational Creativity*, 2016.

[37] J. Gomes, P. Mariano, and A. L. Christensen, "Novelty search in competitive coevolution," in *International Conference on Parallel Problem Solving from Nature*, pp. 233–242, Springer, 2014.

[38] J. Gomes, P. Mariano, and A. L. Christensen, "Novelty-driven cooperative coevolution," *Evolutionary Computation*, vol. 25, no. 2, pp. 275–307, 2017.

[39] K. Sims, "Evolved virtual creatures." https://www.karlsims.com/evolved-virtual-creatures.html. [Online; last accessed 2021-October-13].

[40] D. W. Corne and J. D. Knowles, "Techniques for highly multiobjective optimisation: some nondominated points are better than others," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pp. 773–780, 2007.

[41] P. J. Bentley and J. P. Wakefield, "Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms," in *Soft Computing in Engineering Design and Manufacturing*, pp. 231–240, Springer, 1998.

[42] B. Ross, *Course Notes*. 2021.

[43] S. Luke, "ECJ: A java-based evolutionary computation research system." https://cs.gmu.edu/~eclab/projects/ecj/. [Online; last accessed 2021-October-13].

[44] P. Nadkarni, *Clinical Research Computing: A Practitioner's Handbook*. Academic Press, 2016.

[45] J. A. Brown, "Towards better personas in gaming: Contract based expert systems," in *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 540–541, IEEE, 2015.

[46] R. Ishii, S. Ito, M. Ishihara, T. Harada, and R. Thawonmas, "Monte-carlo tree search implementation of fighting game ais having personas," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8, IEEE, 2018.

[47] A. Liapis, C. Holmgård, G. N. Yannakakis, and J. Togelius, "Procedural personas as critics for dungeon generation," in *European Conference on the Applications of Evolutionary Computation*, pp. 331–343, Springer, 2015.

[48] J. W. Pratt, "Robustness of some procedures for the two-sample location problem," *Journal of the American Statistical Association*, vol. 59, no. 307, pp. 665–680, 1964.

[49] S. Doncieux, A. Laflaquière, and A. Coninx, "Novelty search: a theoretical perspective," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 99–106, 2019.

[50] Z. Li, C. H. Sim, and M. Y. H. Low, "A survey of emergent behavior and its impacts in agent-based systems," in *2006 4th IEEE International Conference on Industrial Informatics*, pp. 1295–1300, IEEE, 2006.

[51] W. B. Langdon, "A many threaded cuda interpreter for genetic programming," in *European Conference on Genetic Programming*, pp. 146–158, Springer, 2010.

[52] M. Maghoumi, "Real-time automatic object classification and tracking using genetic programming and nvidia cuda," *MSc. Dissertation, Brock University*, 2014.

[53] E. F. Anderson, "Off-line evolution of behaviour for autonomous agents in real-time computer games," in *International Conference on Parallel Problem Solving from Nature*, pp. 689–699, Springer, 2002.

[54] A. M. Mora, A. Fernández-Ares, J. J. Merelo, P. García-Sánchez, and C. M. Fernandes, "Effect of noisy fitness in real-time strategy games player behaviour optimisation using evolutionary algorithms," *Journal of Computer Science and Technology*, vol. 27, no. 5, pp. 1007–1023, 2012.

[55] A. E. Eiben, N. Bredeche, M. Hoogendoorn, J. Stradner, J. Timmis, A. M. Tyrrell, and A. Winfield, "The triangle of life: Evolving robots in real-time and real-space," in *Artificial Life Conference Proceedings 13*, pp. 1056–1063, MIT Press, 2013.

[56] Kojima Productions, *Metal Gear Solid V: The Phantom Pain*. Konami, 2015.

[57] Battlestate Games, *Escape from Tarkov*. Battlestate Games, 2017.

[58] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arXiv preprint arXiv:1504.04909*, 2015.

[59] R. Canaan, J. Togelius, A. Nealen, and S. Menzel, "Diverse agents for ad-hoc cooperation in hanabi," in *2019 IEEE Conference on Games (CoG)*, pp. 1–8, IEEE, 2019.

[60] D. Gravina, A. Liapis, and G. Yannakakis, "Surprise search: Beyond objectives and novelty," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 677–684, 2016.

[61] D. Gravina, A. Liapis, and G. N. Yannakakis, "Surprise search for evolutionary divergence," *arXiv preprint arXiv:1706.02556*, 2017.

[62] D. Gravina, A. Liapis, and G. N. Yannakakis, "Quality diversity through surprise," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 603–616, 2018.

[63] R. A. Licitra, Z. D. Hutcheson, E. A. Doucette, and W. E. Dixon, "Single agent herding of n-agents: A switched systems approach," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14374–14379, 2017.

Appendix A

Population Trends

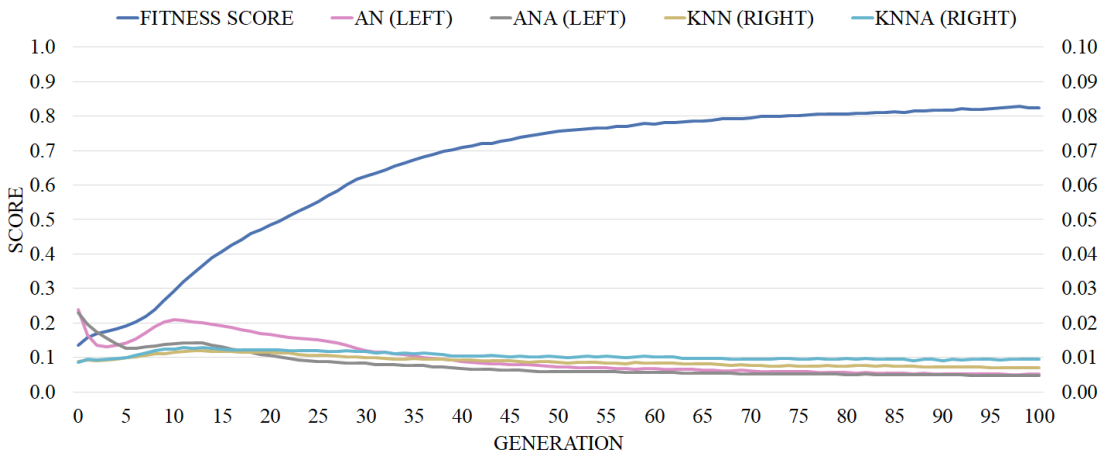


Figure A.1: Series A - Baseline 100F/0D - Population Trend
Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Diversity scores using AN, ANA, KNN, and KNNA are included for later comparisons.

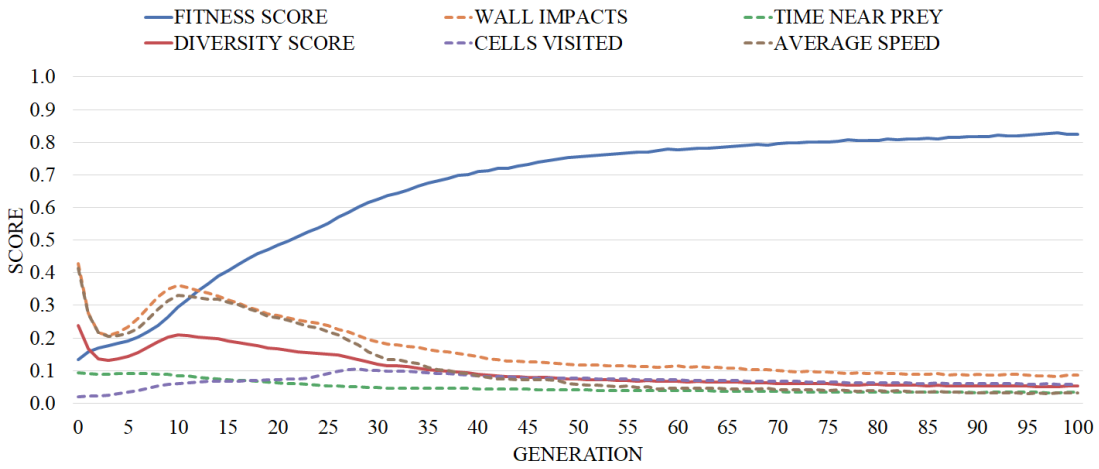


Figure A.2: Series A - AN 100F/0D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

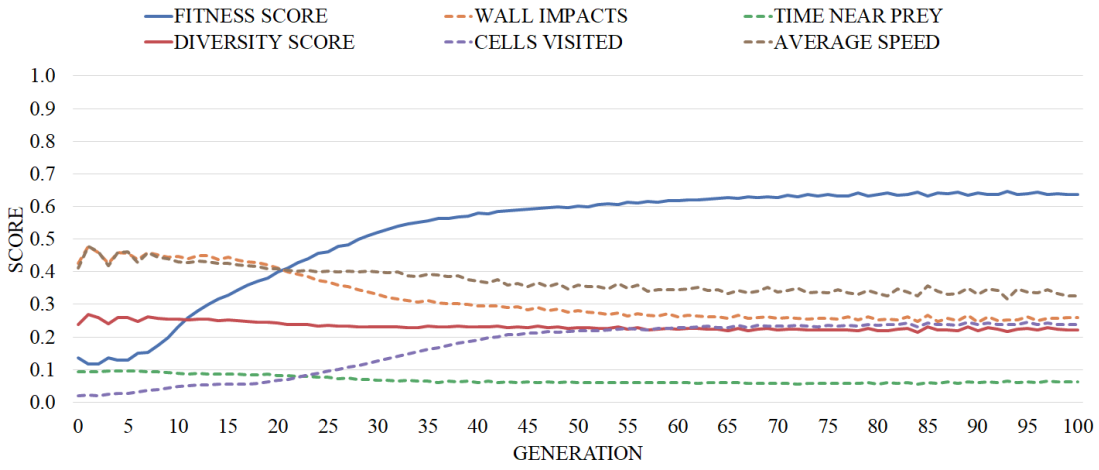


Figure A.3: Series A - AN 50F/50D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

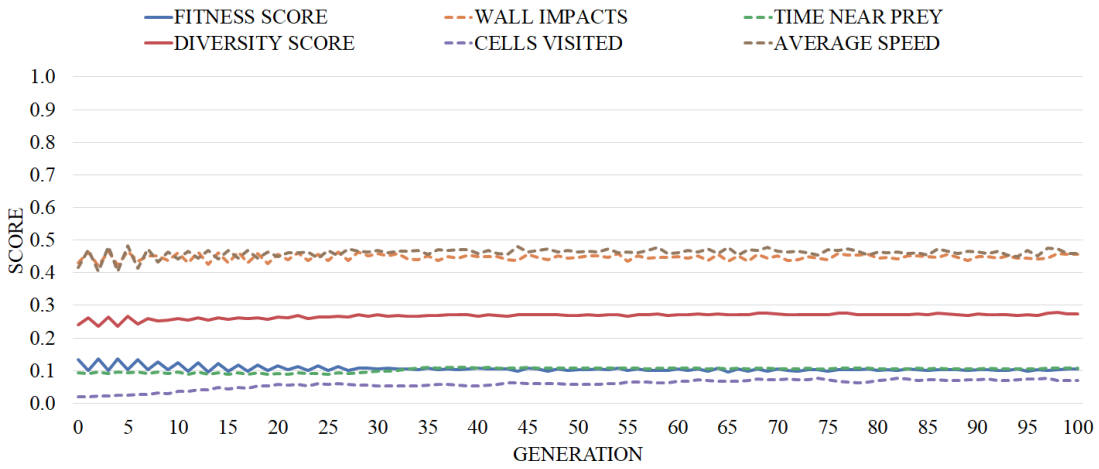


Figure A.4: Series A - AN 0F/100D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

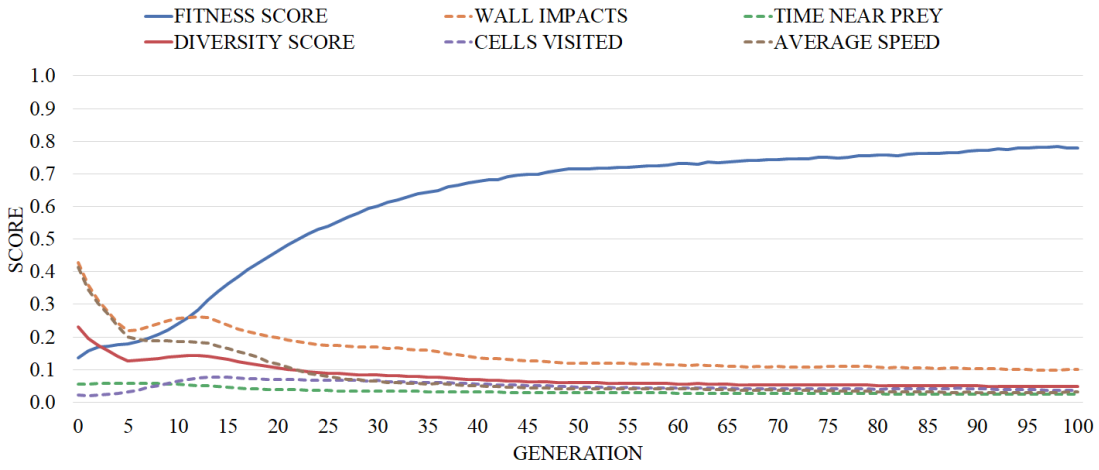


Figure A.5: Series A - ANA 100F/0D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

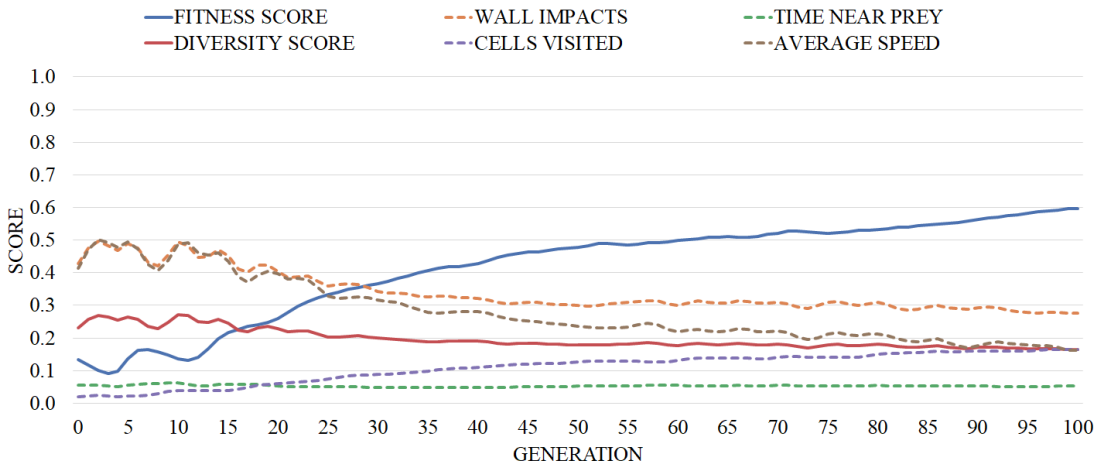


Figure A.6: Series A - ANA 50F/50D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

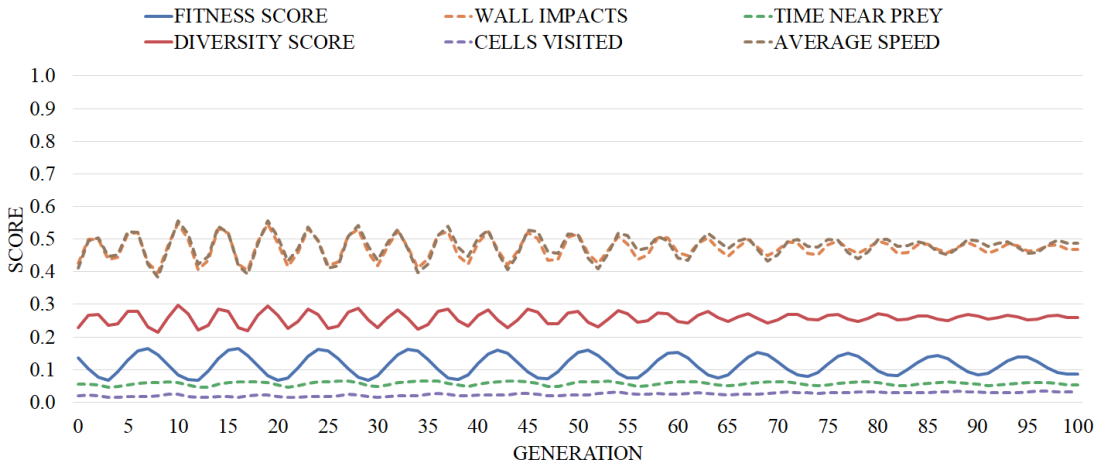


Figure A.7: Series A - ANA 0F/100D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

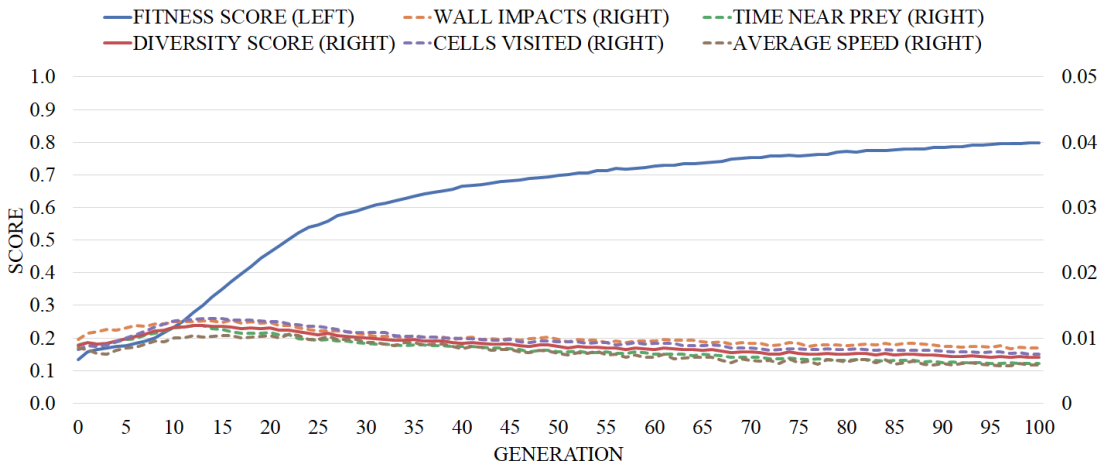


Figure A.8: Series A - KNN 100F/0D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

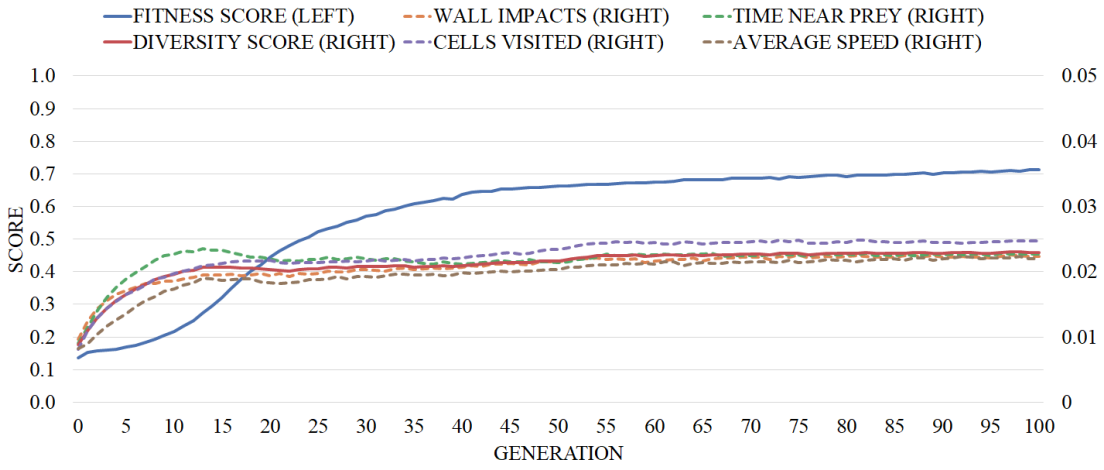


Figure A.9: Series A - KNN 50F/50D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

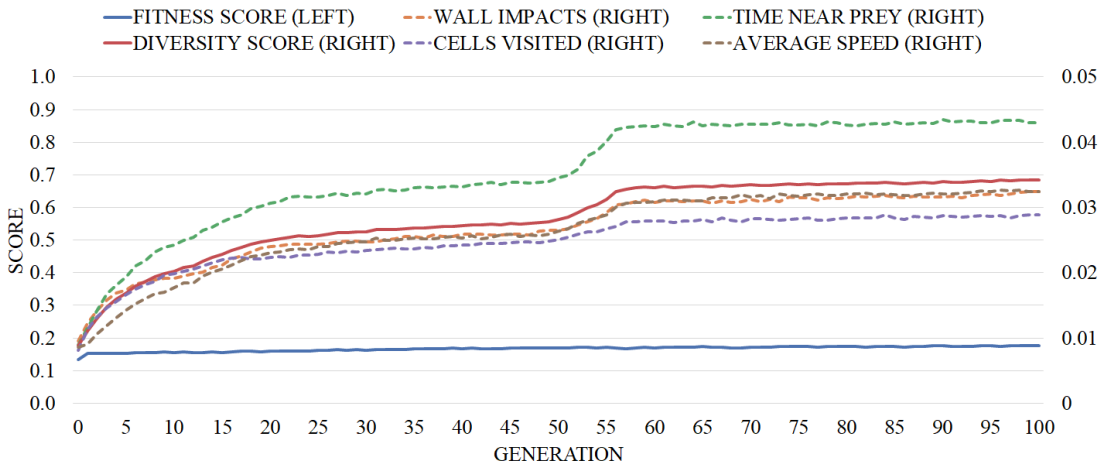


Figure A.10: Series A - KNN 0F/100D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

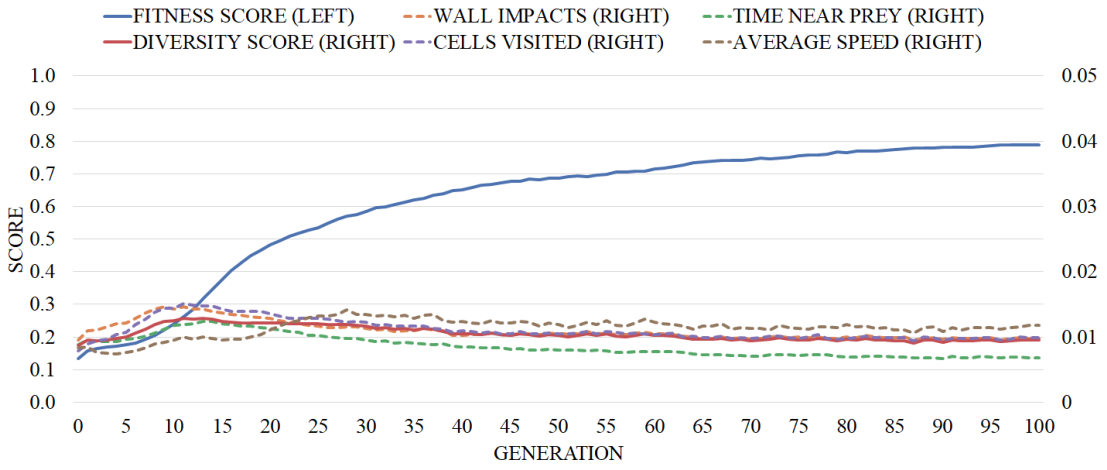


Figure A.11: Series A - KNNA 100F/0D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

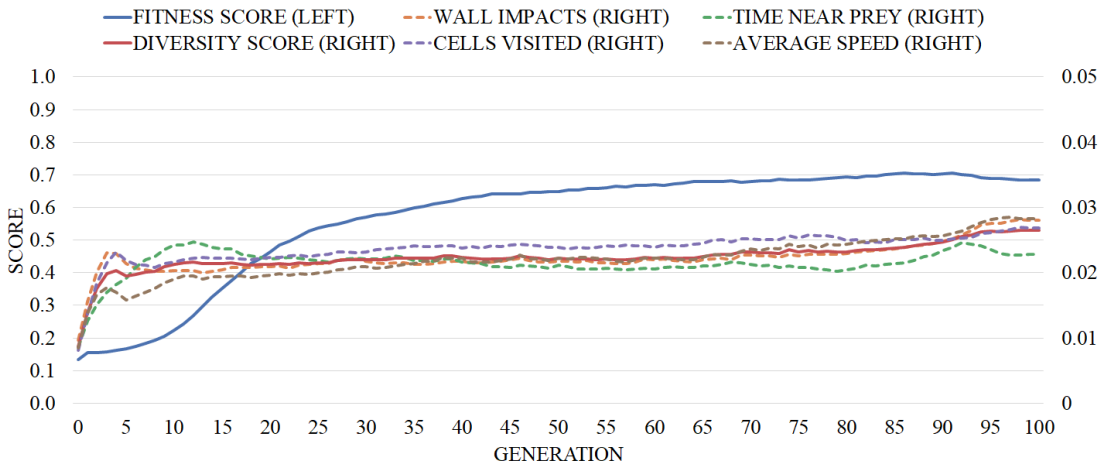


Figure A.12: Series A - KNNA 50F/50D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

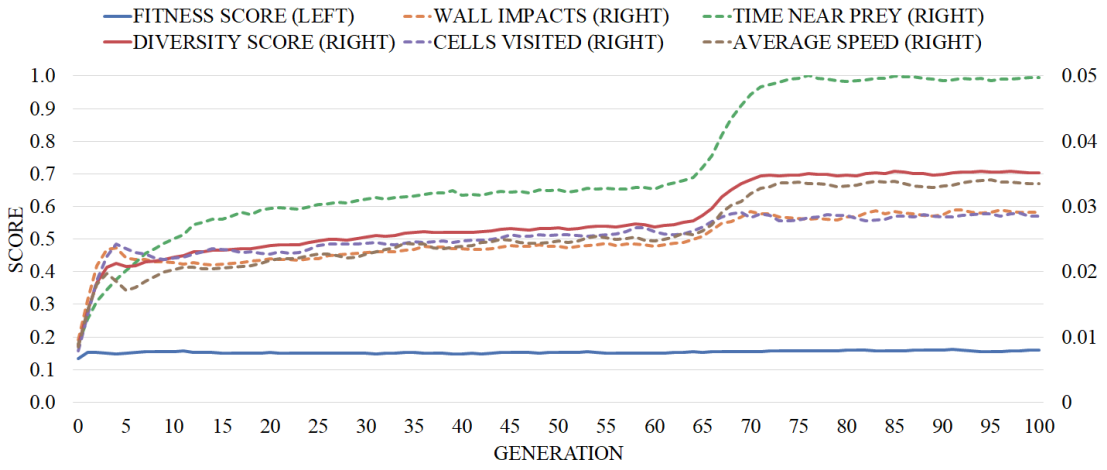


Figure A.13: Series A - KNNA 0F/100D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

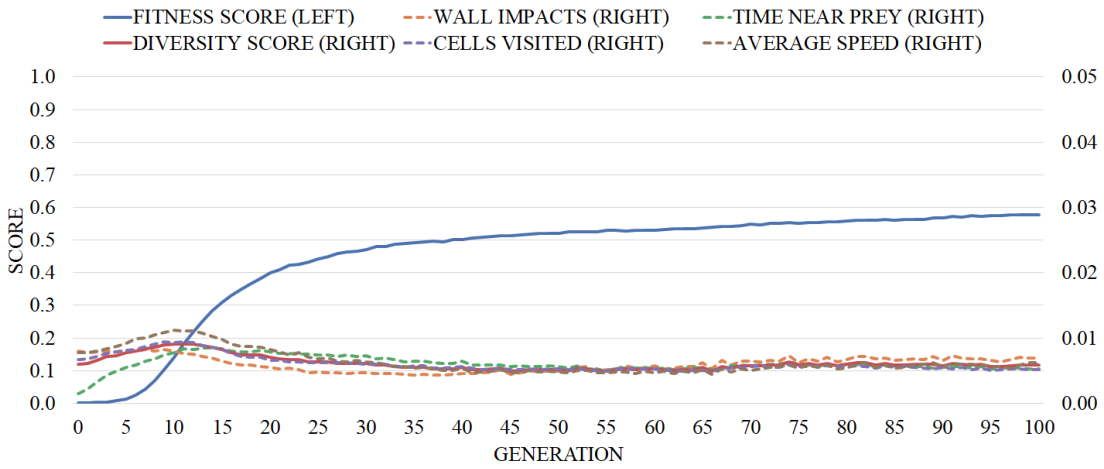


Figure A.14: Series B - KNNA 100F/0D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

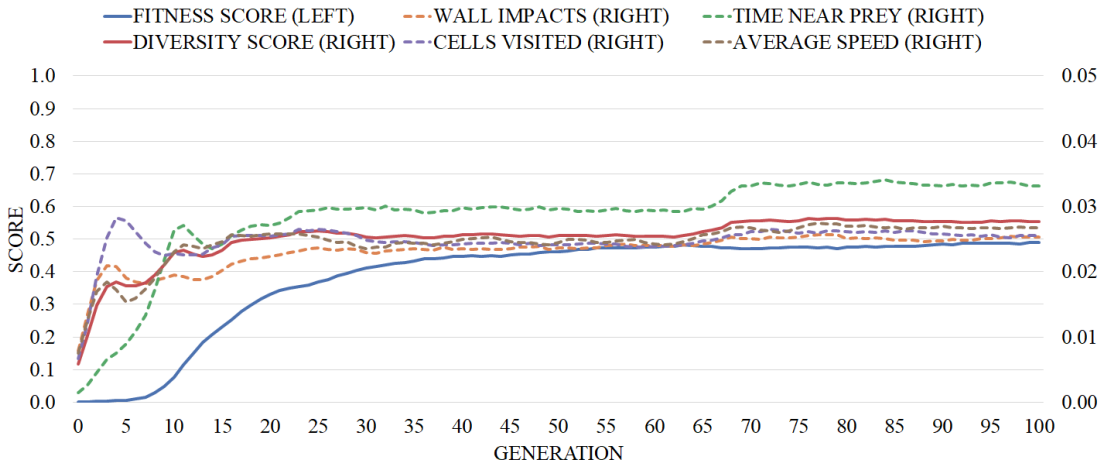


Figure A.15: Series B - KNNA 50F/50D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

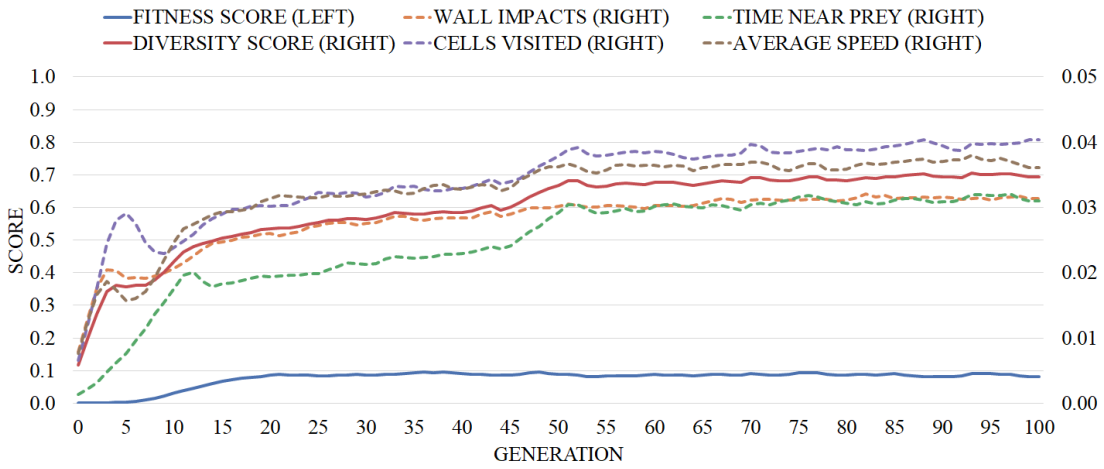


Figure A.16: Series B - KNNA 0F/100D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

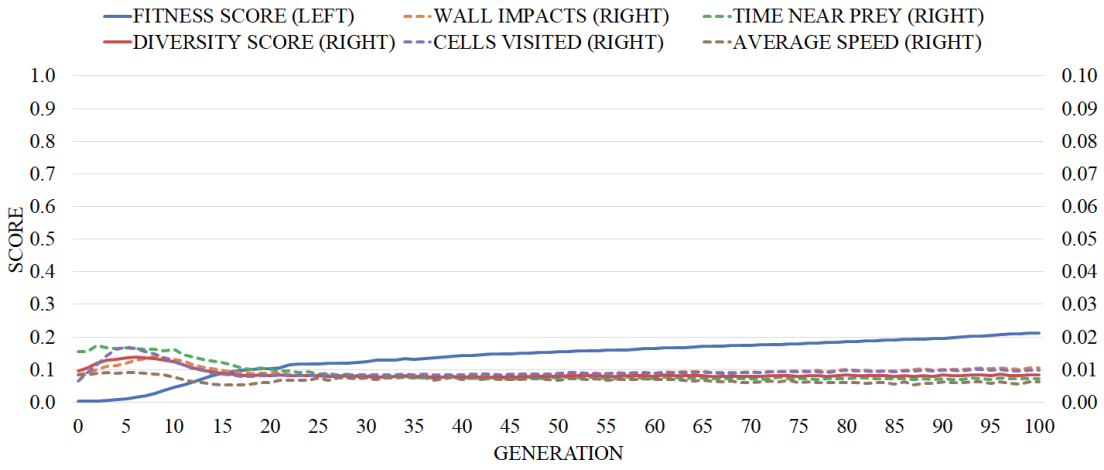


Figure A.17: Series C - KNNA 100F/0D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

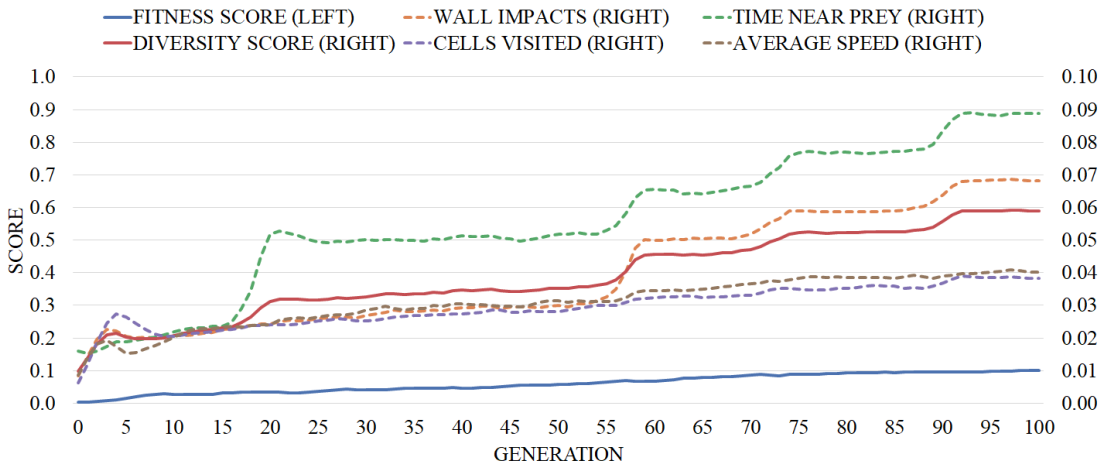


Figure A.18: Series C - KNNA 50F/50D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

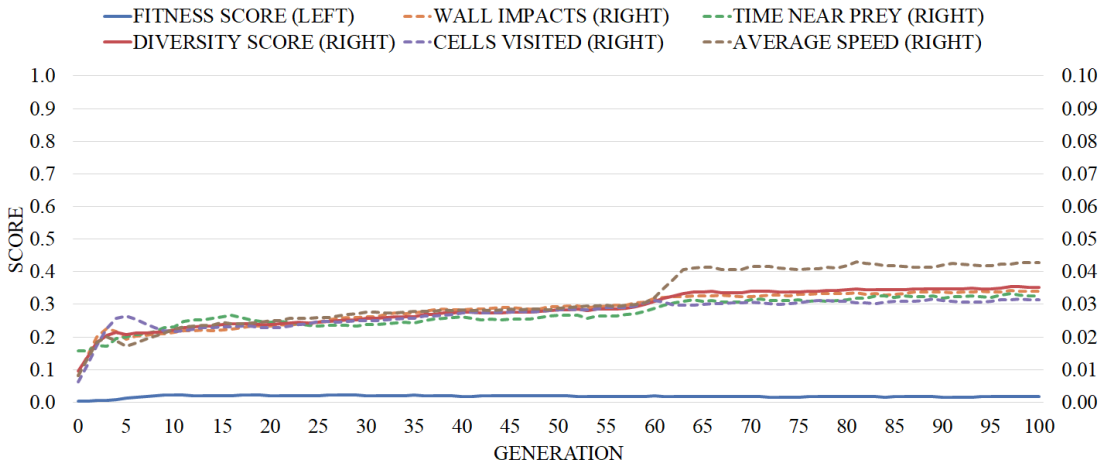


Figure A.19: Series C - KNNA 0F/100D - Population Trend

Population trend of 1000 individuals over 100 generations, averaged over 30 separate runs. Solid lines represent scores. Dashed lines represent behaviour distances.

Appendix B

Final Scores

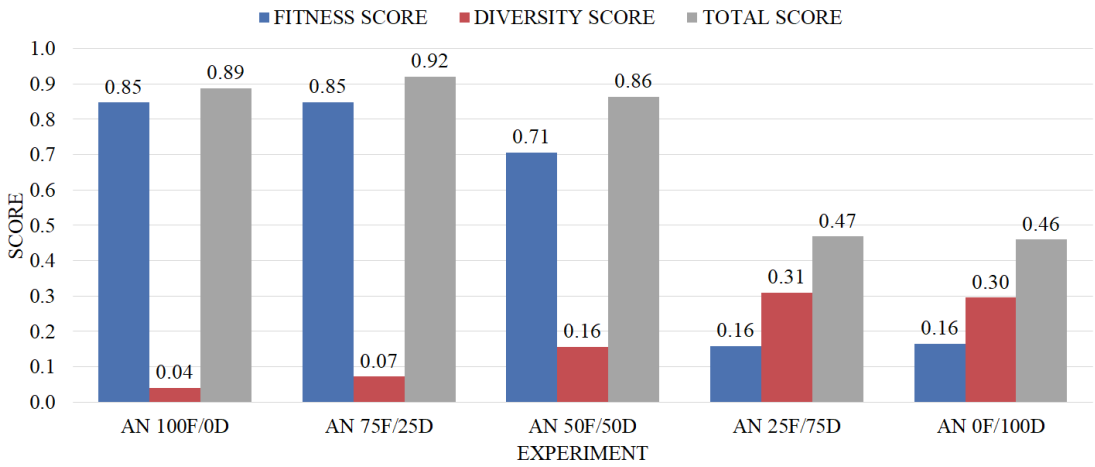


Figure B.1: Series A - AN - Final Scores

Average scores of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations. Total score is supplementary and was not used by the GP.

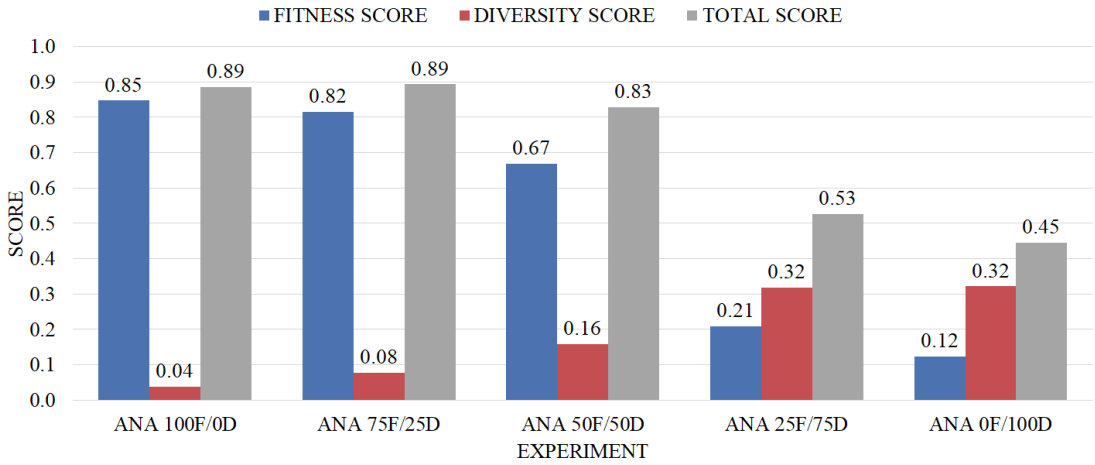


Figure B.2: Series A - ANA - Final Scores

Average scores of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations. Total score is supplementary and was not used by the GP.

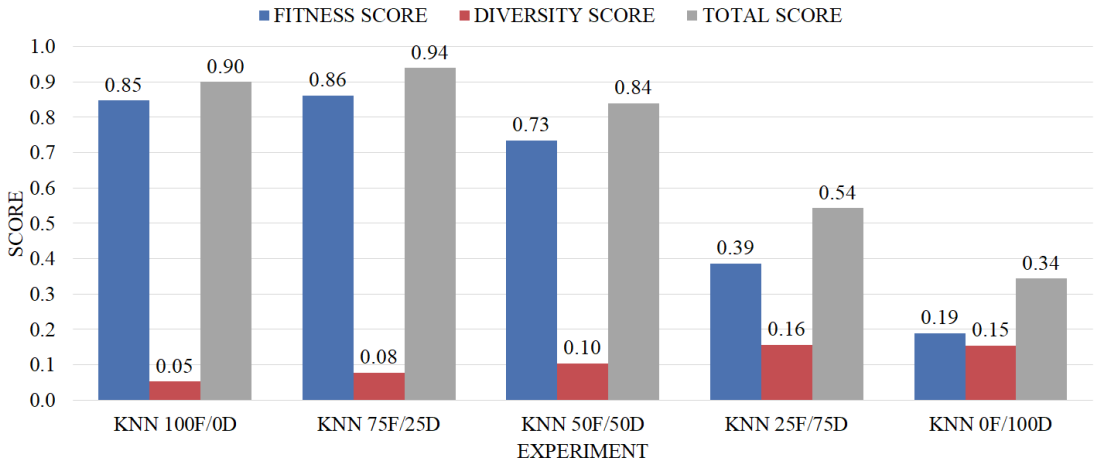


Figure B.3: Series A - KNN - Final Scores

Average scores of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations. Total score is supplementary and was not used by the GP.

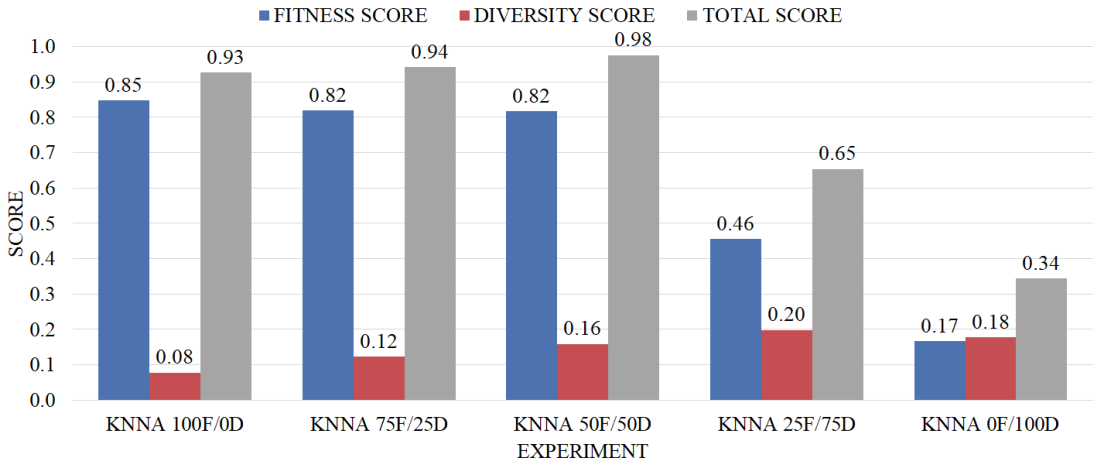


Figure B.4: Series A - KNNA - Final Scores

Average scores of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations. Total score is supplementary and was not used by the GP.

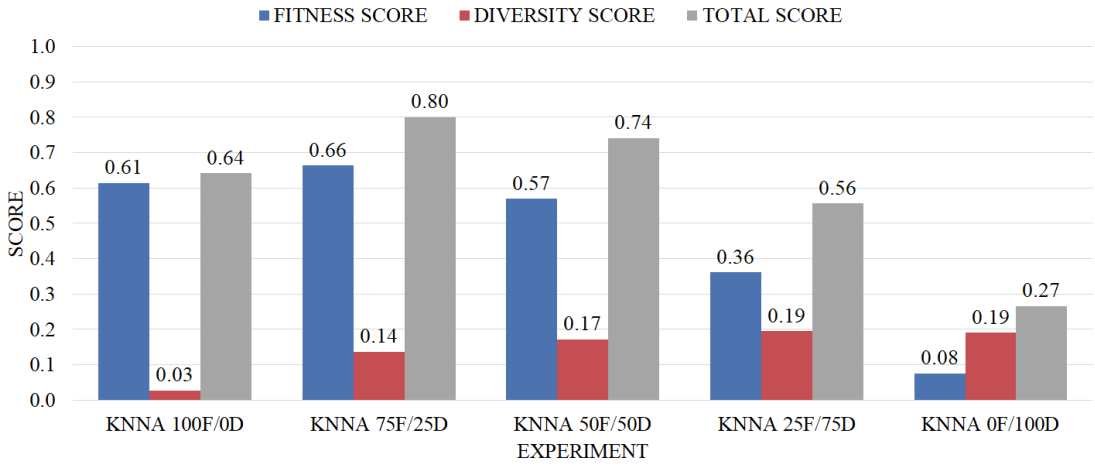


Figure B.5: Series B - KNNA - Final Scores

Average scores of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations. Total score is supplementary and was not used by the GP.

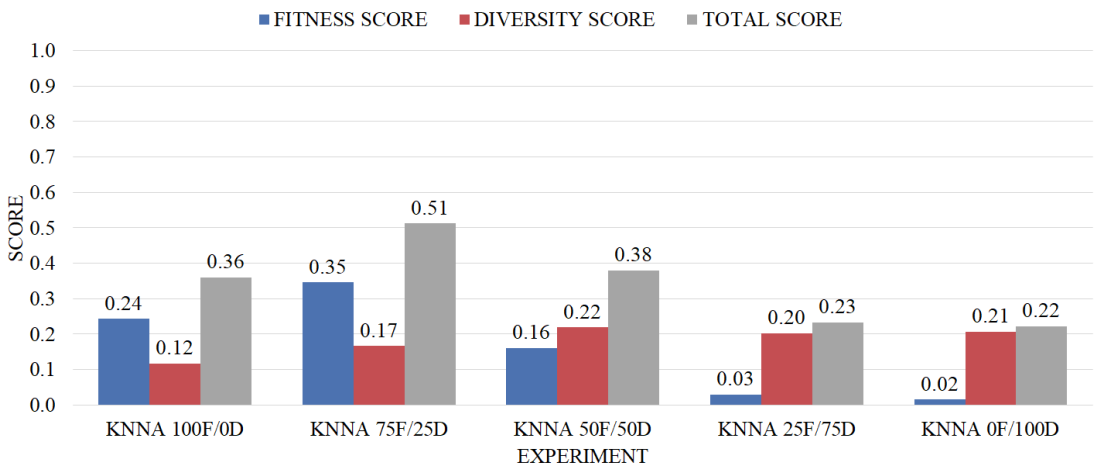


Figure B.6: Series C - KNNA - Final Scores

Average scores of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations. Total score is supplementary and was not used by the GP.

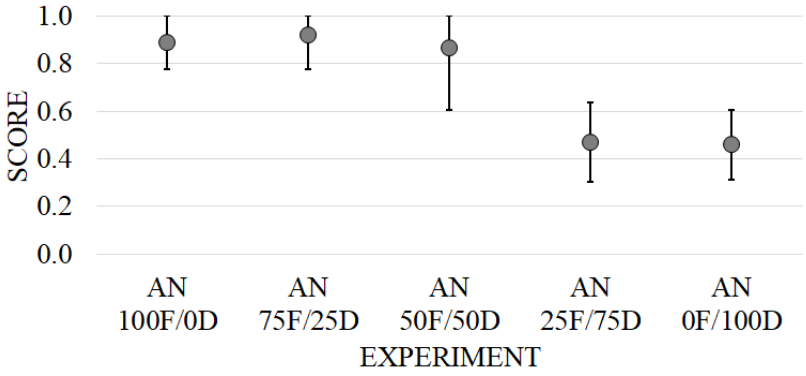


Figure B.7: Series A - AN - Deviation

Average total score with standard deviation of the best individual from each of the 30 runs used in each experiment.

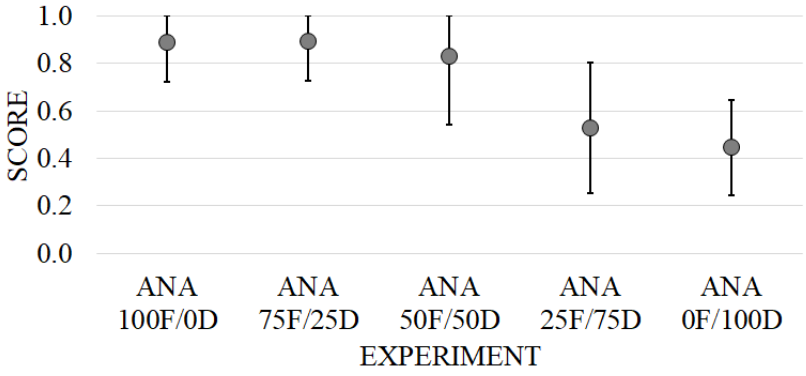


Figure B.8: Series A - ANA - Deviation

Average total score with standard deviation of the best individual from each of the 30 runs used in each experiment.

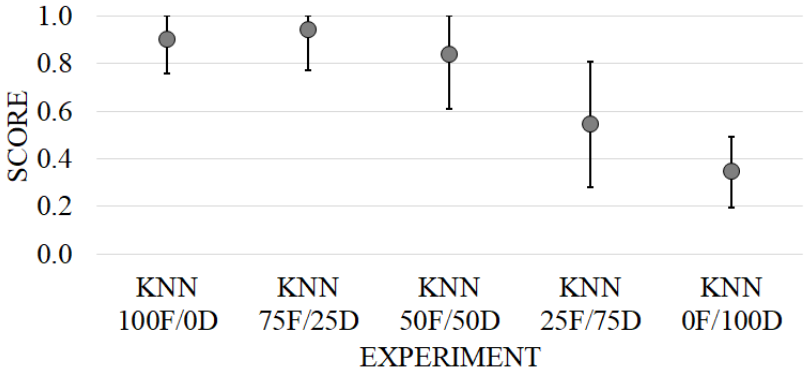


Figure B.9: Series A - KNN - Deviation
Average total score with standard deviation of the best individual from each of the 30 runs used in each experiment.

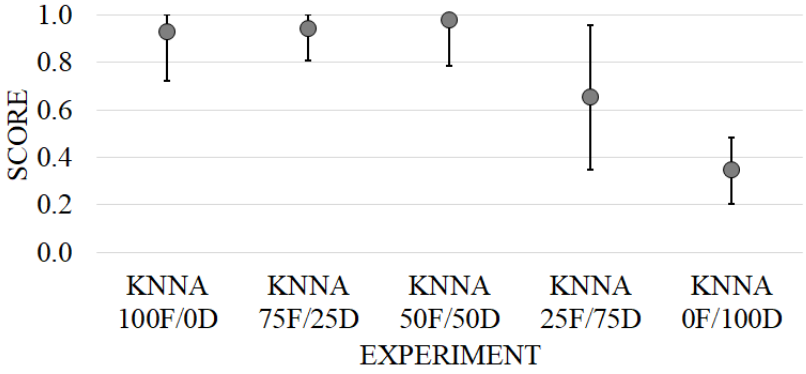


Figure B.10: Series A - KNNA - Deviation
Average total score with standard deviation of the best individual from each of the 30 runs used in each experiment.

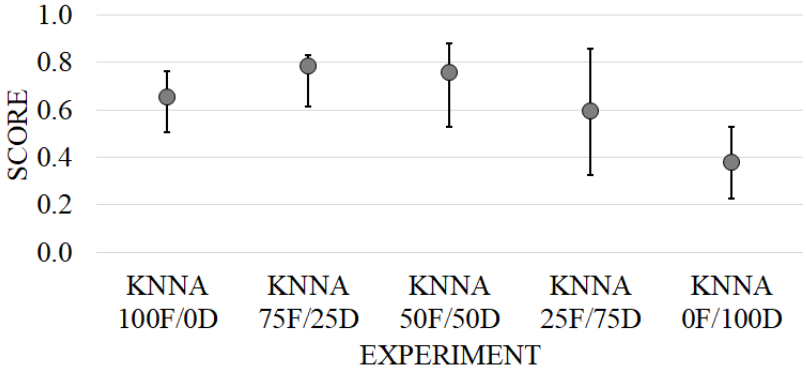


Figure B.11: Series B - KNNA - Deviation
Average total score with standard deviation of the best individual from each of the 30 runs used in each experiment.

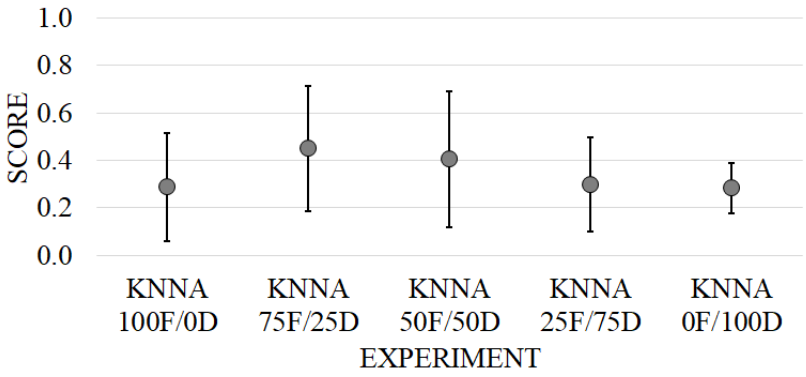


Figure B.12: Series C - KNNA - Deviation

Average total score with standard deviation of the best individual from each of the 30 runs used in each experiment.

Appendix C

Quantitative Behaviours

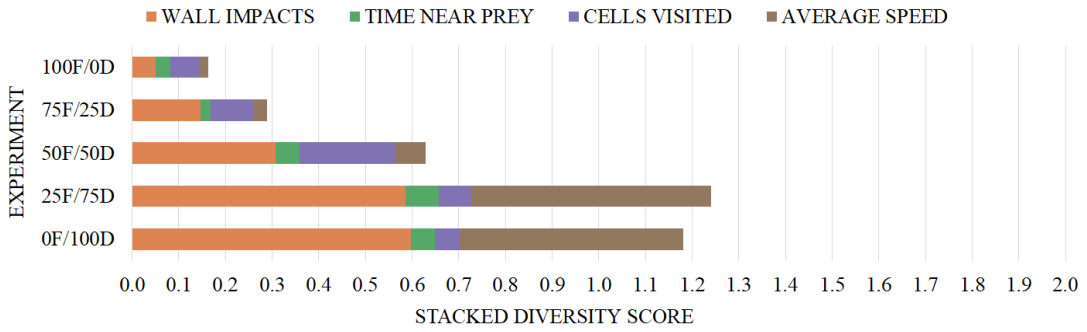


Figure C.1: Series A - AN - Quantitative Behaviours

Average quantitative behaviour distances of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations.

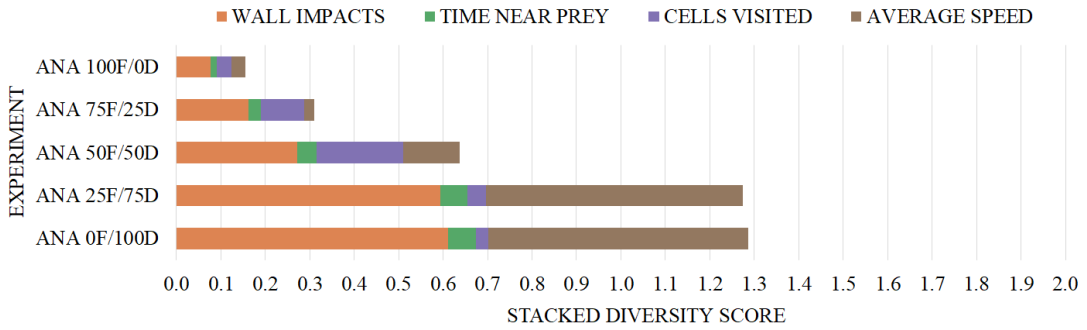


Figure C.2: Series A - ANA - Quantitative Behaviours

Average quantitative behaviour distances of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations.

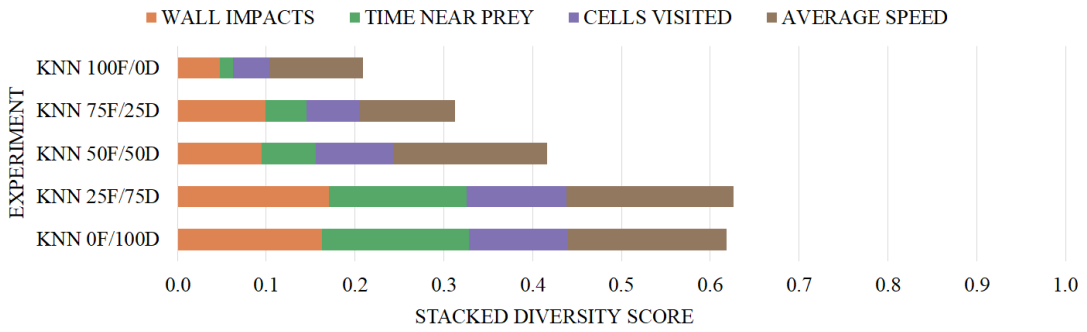


Figure C.3: Series A - KNN - Quantitative Behaviours

Average quantitative behaviour distances of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations.

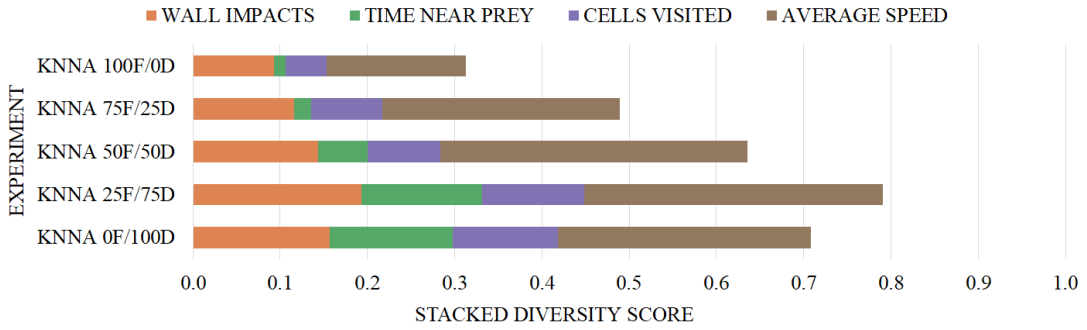


Figure C.4: Series A - KNNA - Quantitative Behaviours

Average quantitative behaviour distances of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations.

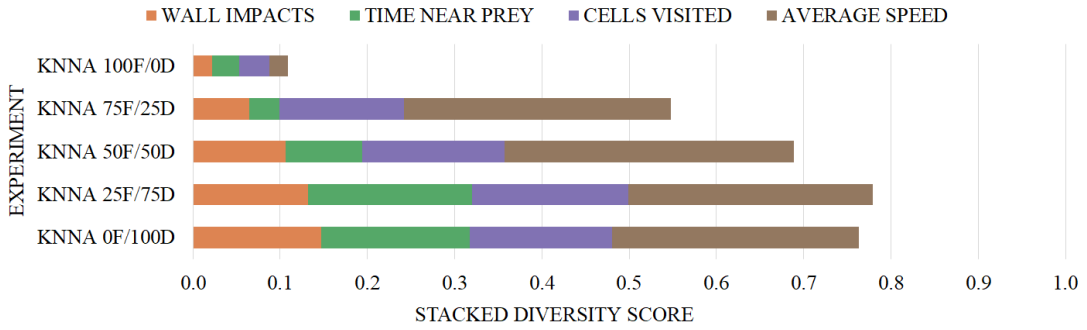


Figure C.5: Series B - KNNA - Quantitative Behaviours

Average quantitative behaviour distances of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations.

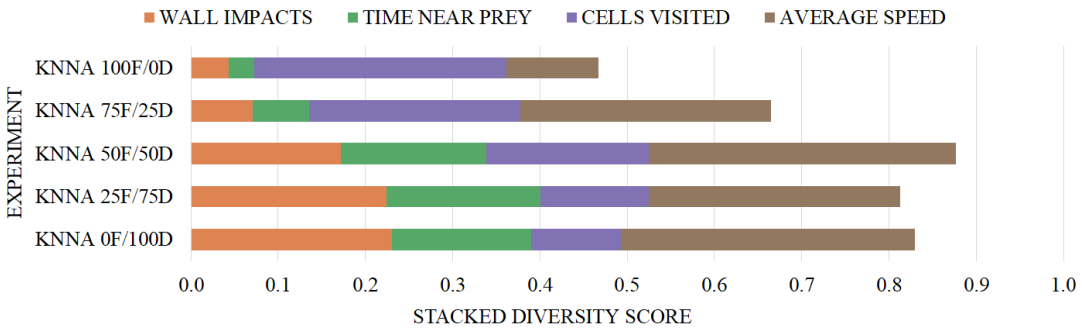


Figure C.6: Series C - KNNA - Quantitative Behaviours

Average quantitative behaviour distances of the best individual from each of the 30 runs used in each experiment. Each individual is tested and averaged over five simulations.

Appendix D

Qualitative Behaviours

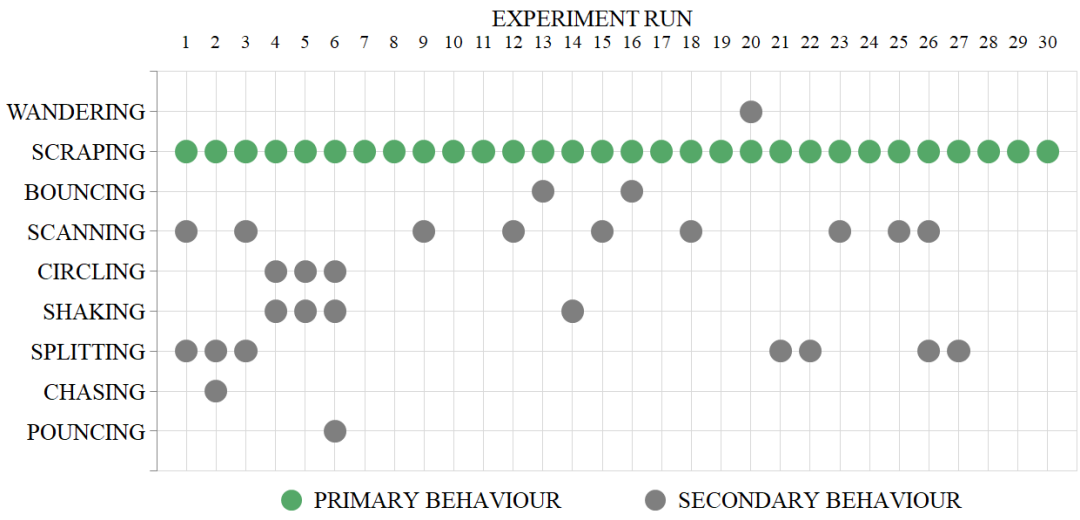


Figure D.1: Series A - 100F/0D - Qualitative Behaviours
Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation.

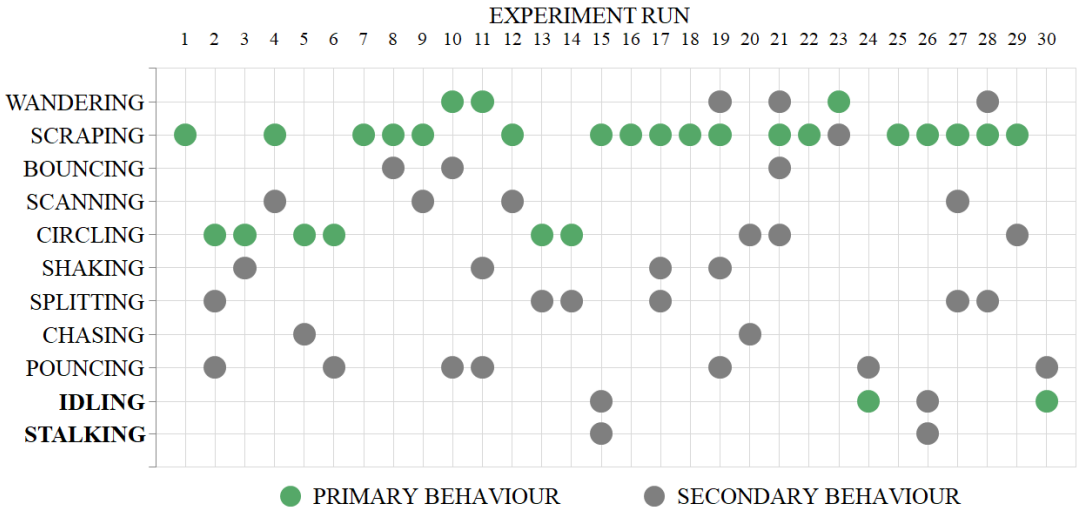


Figure D.2: Series A - AN 50F/50D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

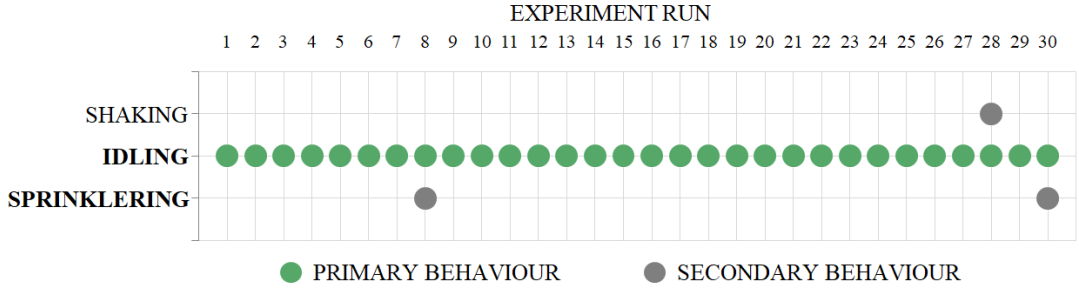


Figure D.3: Series A - AN 0F/100D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

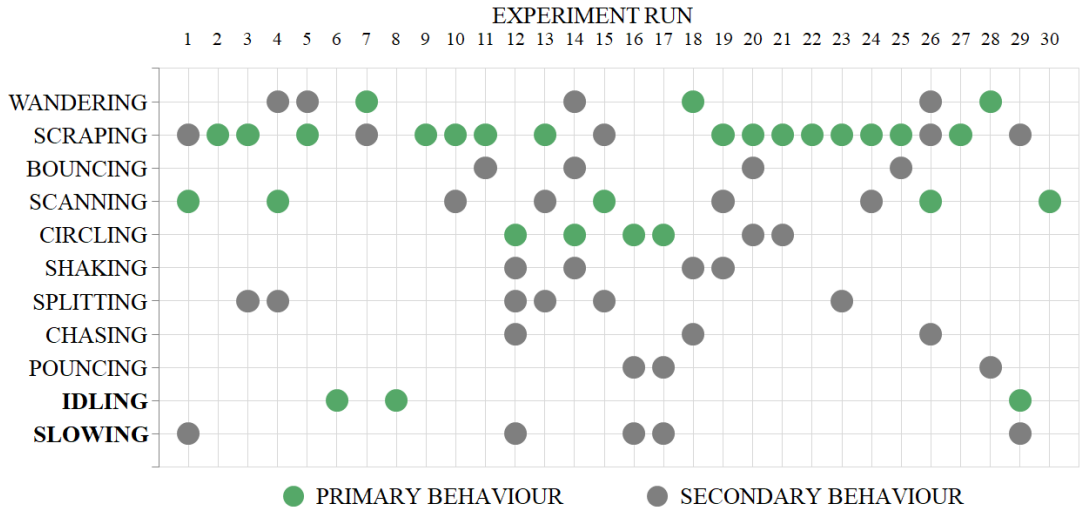


Figure D.4: Series A - ANA 50F/50D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

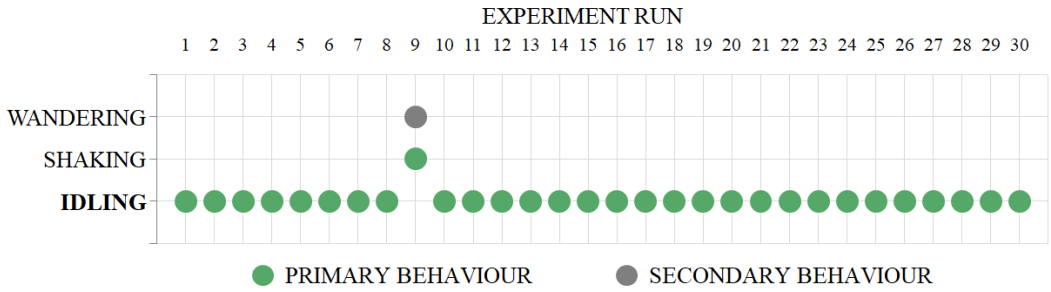


Figure D.5: Series A - ANA 0F/100D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

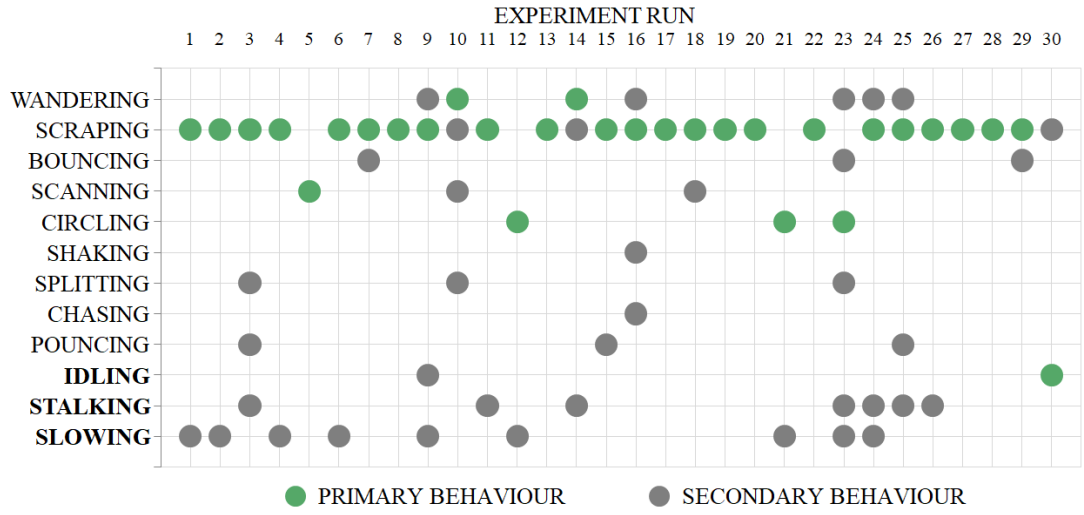


Figure D.6: Series A - KNN 50F/50D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

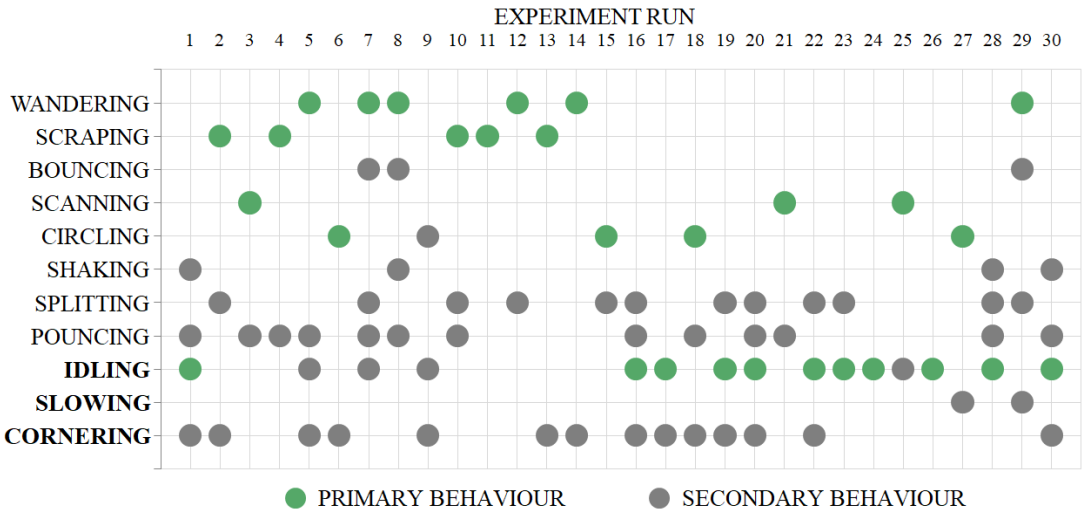


Figure D.7: Series A - KNN 0F/100D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

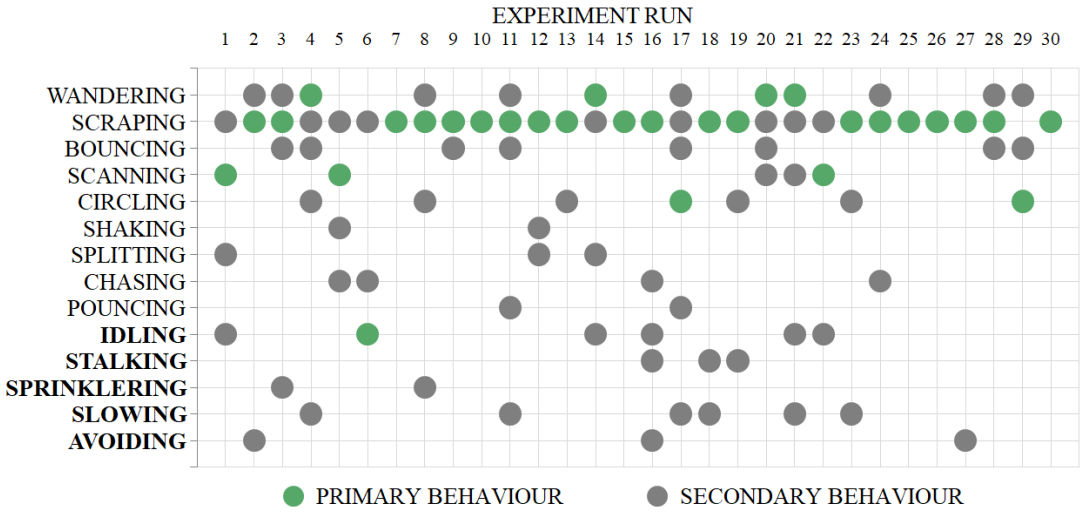


Figure D.8: Series A - KNNA 50F/50D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

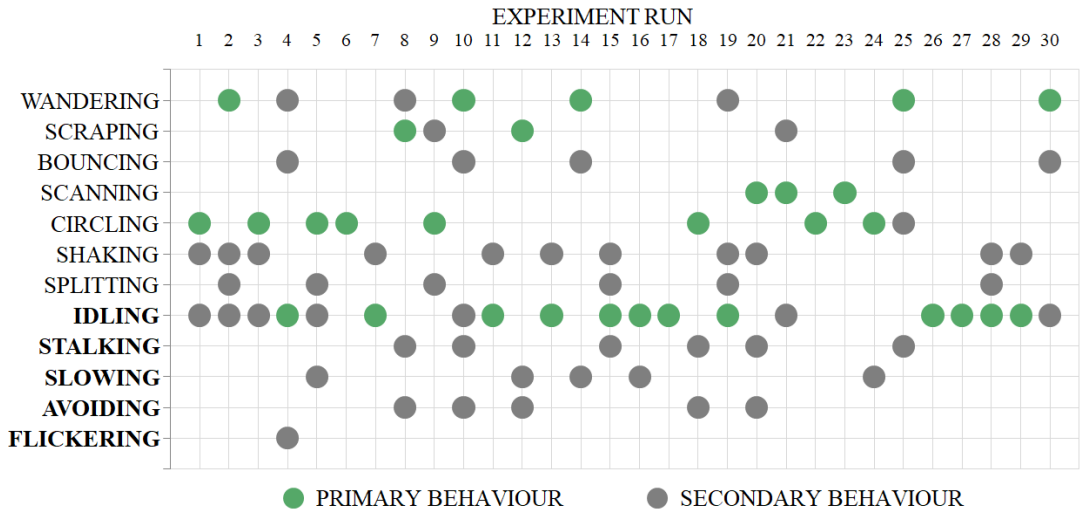


Figure D.9: Series A - KNNA 0F/100D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

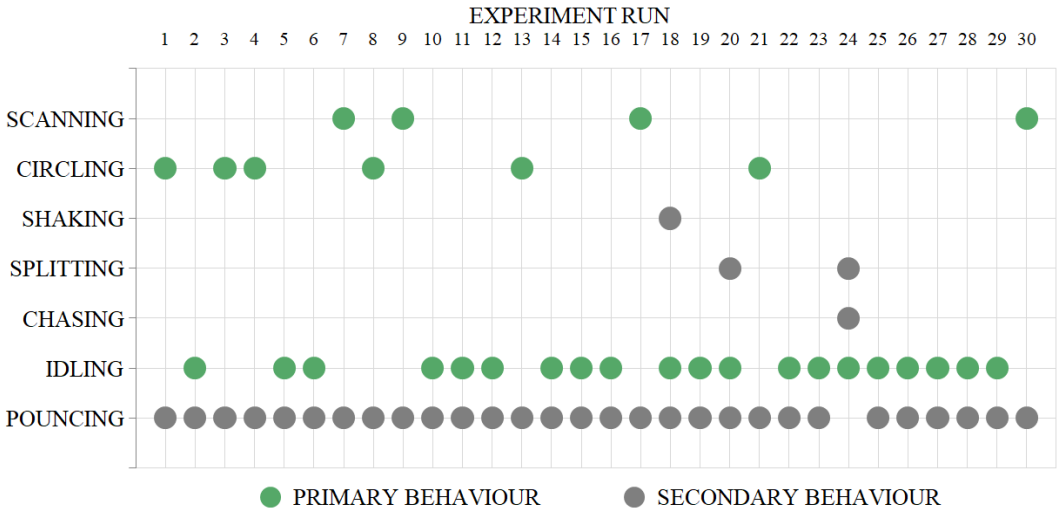


Figure D.10: Series B - 100F/0D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation.

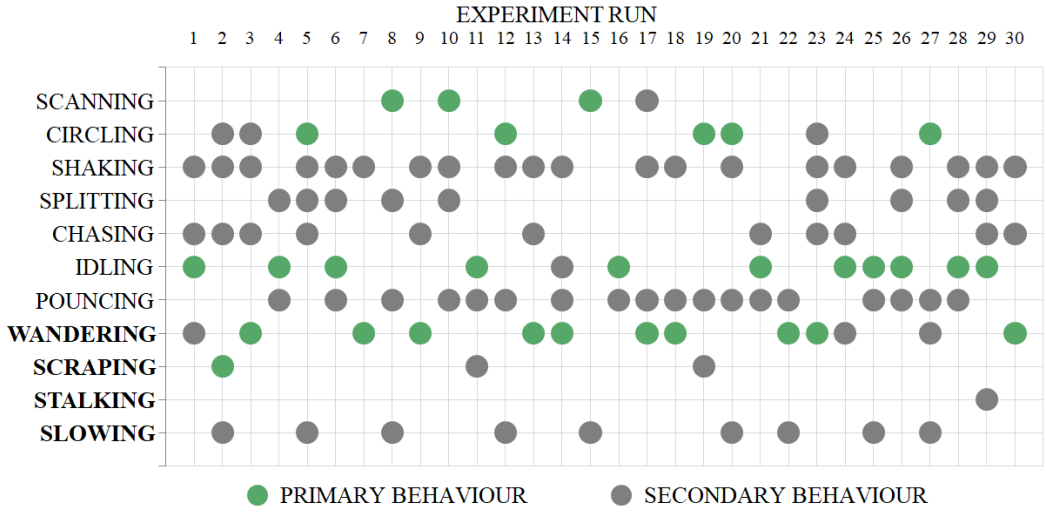


Figure D.11: Series B - KNNA 50F/50D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

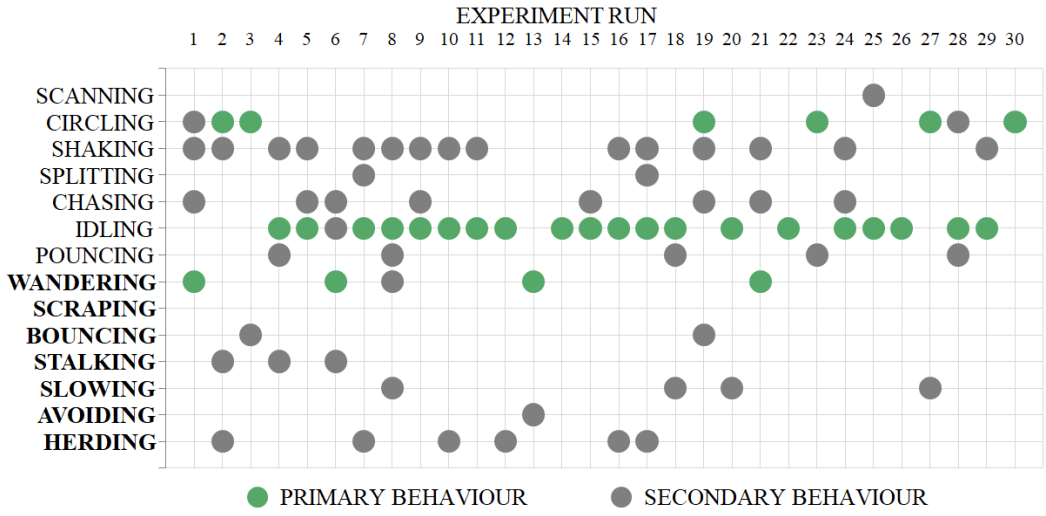


Figure D.12: Series B - KNNA 0F/100D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

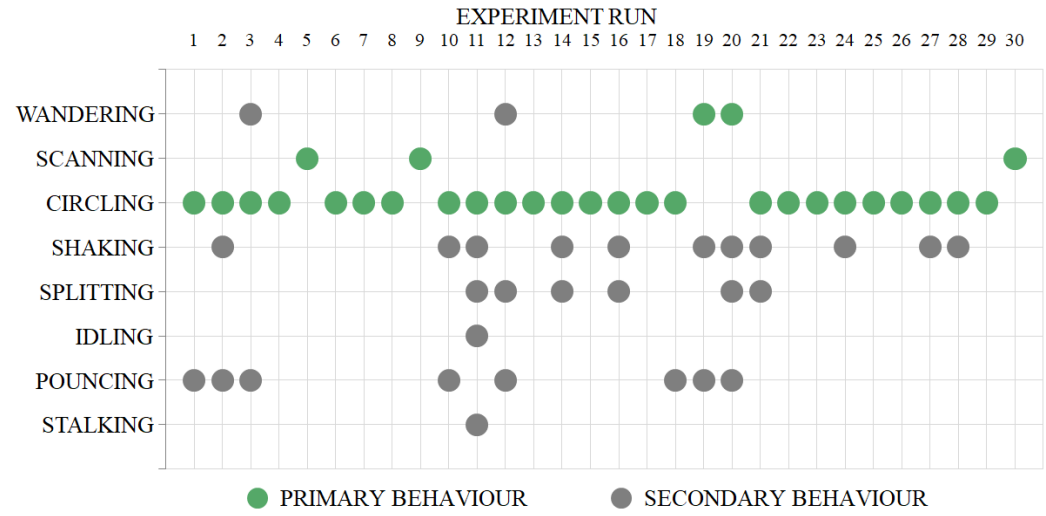


Figure D.13: Series C - 100F/0D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation.

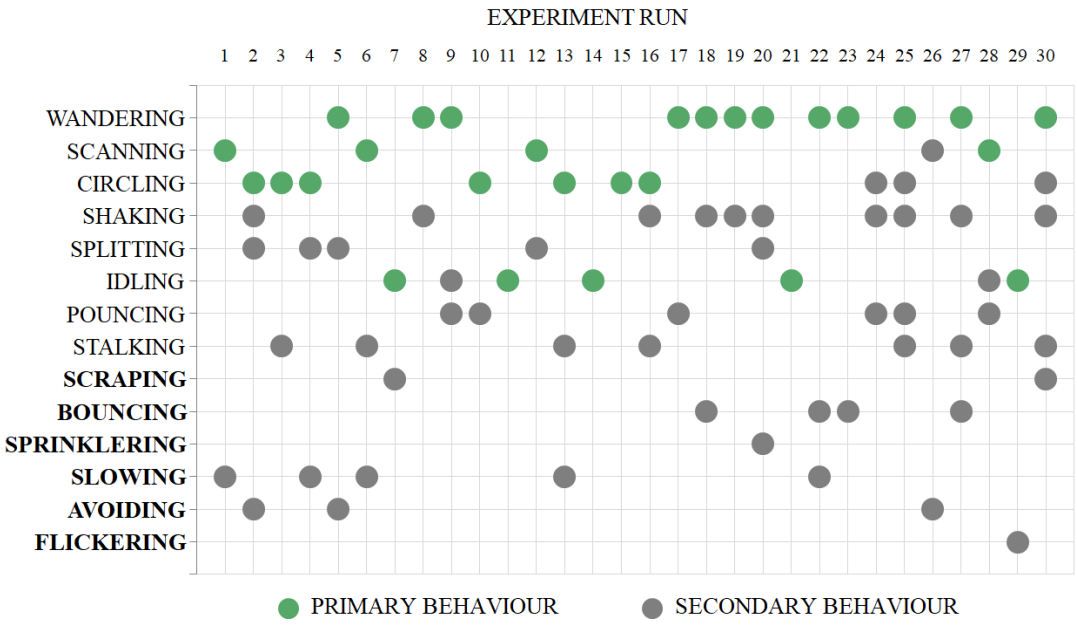


Figure D.14: Series C - KNNA 50F/50D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

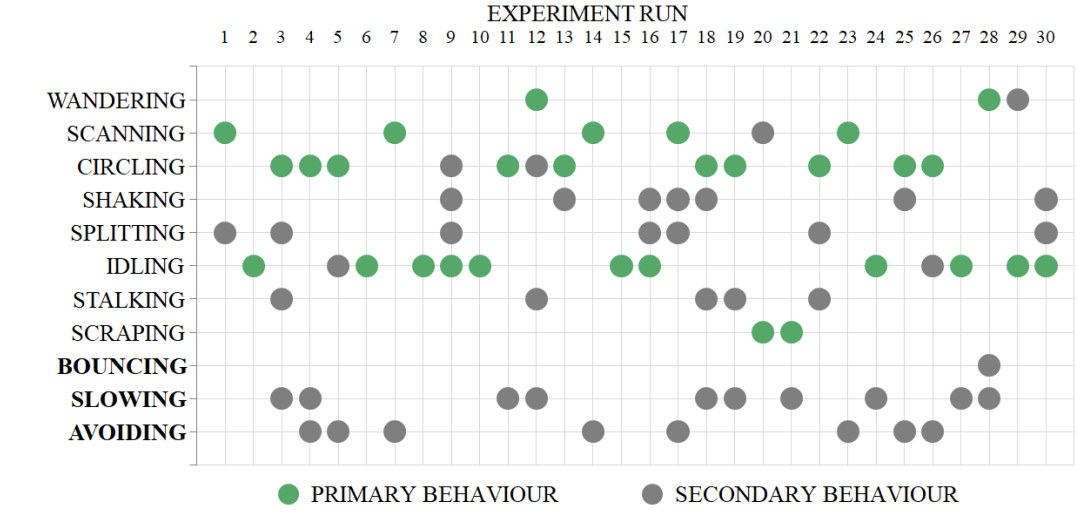


Figure D.15: Series C - KNNA 0F/100D - Qualitative Behaviours

Empirical analysis of the best solution from each of the 30 runs used in this experiment. Tallies indicate that the behaviour was observed at least once during that simulation. Bold represents new behaviours (emerged with diversity but did not exist during baseline).

Appendix E

Programs

```
(Rotate (* (MaxXY (- (Normalize (Rotate (BreakX
(VBranch HittingWall VEphemeral(0.1163201749252728,0.8345827209726462)
    (ReplaceX FGet PredRotation))) (ReplaceY (VectorLength
(VInvert VEphemeral(0.8631007189948338,0.06438519257808153)))
(- (Normalize (VInvert (Rotate (* (/ (MaxXY
    (MakeVector (FSet FGet) (FBranch CEphemeral(0.057331931849001094)
        FEphemeral(0.034126720545868294) StepsSinceLastCatch)))
    (Dot VSightCheck (Scale StepsSinceLastCatch (- PredLocation
        VProximityCheck)))) (VectorLength (MakeVector
    StepsSinceLastCatch (VectorLength (VInvert VProximityCheck)))))
    (Scale (Average (FBranch CEphemeral(0.057331931849001094)
        FEphemeral(0.034126720545868294) StepsSinceLastCatch) (Dot
        PredRotation VGet)) (Normalize (- (+ PredLocation (-
        PredLocation VProximityCheck)) (VInvert VGet)))))))
    (Scale (FSet StepsSinceLastCatch) (Normalize (VSet
        VGet))))))) (- PredLocation VProximityCheck)))
(Sin FGet)) (SwapXY (VBranch (CSet (ToCondition
FEphemeral(0.6605680173147813))) (ReplaceX StepsSinceLastCatch
(ReplaceY (- (FBranch (ToCondition (Dot VSightCheck
    (Scale (MaxXY (ReplaceX FGet PredRotation)) (VInvert
        PredLocation)))) (Average (FBranch CEphemeral(0.06408813274623909)
    FEphemeral(0.04809906001807422) StepsSinceLastCatch) (Sin
    (+ (FBranch CSightCheck StepsSinceLastCatch FGet) StepsSinceLastCatch)))
    (/ StepsSinceLastCatch (Dot PredRotation VGet))) (/ FEphemeral(0.8720)
    (FBranch (ToCondition (BreakX (VInvert (- PredRotation
        PredRotation)))) (Average (Dot VSightCheck (ReplaceX FGet
        PredRotation)) (Average (FBranch CEphemeral(0.057331931849001094)
        FEphemeral(0.034126720545868294) StepsSinceLastCatch) (/
        (VectorLength (VInvert VProximityCheck)) (+
        StepsSinceLastCatch FGet)))) (/ (MinXY (- PredLocation VProximityCheck))
        (FSet FGet))))) (Normalize (ReplaceY (MaxXY (VInvert
    (- PredRotation PredRotation))) (Scale (VectorLength
    (MakeVector StepsSinceLastCatch (FInvert (BreakX (-
        PredLocation (ReplaceX FEphemeral(0.17136268779239172) VSightCheck))))))
    (VBranch CProxCheck VProximityCheck PredRotation))))))
(Normalize (ReplaceY (- (Dot PredRotation VGet) StepsSinceLastCatch)
    (Normalize (ReplaceY (MaxXY (VInvert (- PredLocation
        VProximityCheck))) (VInvert PredLocation))))))))
```

Figure E.1: Full Genetic Program - 100F/0D

The code for the best individual of run 01 for 100F/0D.
Corresponds to the individual in run 01 in Figure D.1

```
(ReplaceY (VectorLength (VBranch HittingWall
     PredRotation (+ VEphemeral(0.0031,0.1687)
     VSightCheck))) (Scale (FBranch (CSet (> (-
     (FInvert (FSet FGet)) (FInvert (Dot VGet (Normalize
     (SwapXY (ReplaceY FEphemeral(0.39355955085149197) VSightCheck))))))
     (FBranch (CSet (> (FBranch (CSet (> FGet
         (VectorLength (VBranch HittingWall PredRotation
             VGet)))) (- (Dot VEphemeral(0.6921,0.8460)
         VProximityCheck) (- (FInvert StepsSinceLastCatch) (Dot
         (- (ReplaceX (VectorLength (VBranch HittingWall
             PredRotation VGet)) (VBranch (CSet CSightCheck)
             (+ VEphemeral(0.00318568899480165,0.1687335539572502)
                 VSightCheck) (VInvert PredRotation))) VEphemeral(0.2531,3.57705E-4))
         VSightCheck))) (MinXY (- VGet VEphemeral(0.2531,3.5770E-4))))
         (FBranch (CSet (> FGet (VectorLength (VInvert
             PredRotation)))) (Dot VSightCheck VSightCheck)
             (MinXY (- (SwapXY (SwapXY (VSet (+ VProximityCheck
                 VGet)))) VEphemeral(0.25312640630028027,3.5770E-4))))))
         (- (Sin (FSet (Average (MaxXY PredLocation) (* FEphemeral(0.3621)
             StepsSinceLastCatch)))) (- (- (FInvert (* (FInvert (MinXY
             (Normalize (VInvert VSightCheck)))) (VectorLength
             VEphemeral(0.4524764247818722,0.8066496303777747))))
             (Dot VSightCheck (+ (MakeVector FEphemeral(0.45281033345939836)
                 (Dot VGet VSightCheck)) (+ (SwapXY VProximityCheck)
                 VEphemeral(0.35333087982965106,0.13453288258312113)))))
             (FInvert (MinXY (ReplaceX (VectorLength (VBranch
                 HittingWall PredRotation (VBranch (CSet CSightCheck)
                 (Rotate StepsSinceLastCatch VSightCheck) VSightCheck)))
                 (Normalize VSightCheck)))))) (MinXY (- (ReplaceX
         (FInvert (FInvert (Dot VSightCheck VProximityCheck)))
         (VBranch (CSet CSightCheck) (Normalize (Rotate
             (VectorLength (VInvert PredRotation)) PredRotation)) (VBranch
             HittingWall PredRotation VGet))) VEphemeral(0.2531,3.57705E-4))))))
     (- (FInvert (MinXY (SwapXY (- (VBranch HittingWall
         PredRotation VGet) (MakeVector FEphemeral(0.45281033345939836)
         (MinXY (VBranch HittingWall VEphemeral(0.2531,3.57705E-4)
             VGet)))))) (- (FBranch (CSet (> (Sin FEphemeral(0.4343))

  ...
```

Figure E.2: Full Genetic Program - KNNA 50F/50D - Part 1

The first half of the code for the best individual of run 01 for KNNA 50F/50D. Corresponds to the individual in run 01 in Figure D.8

```
...

(- FGet (- (* (- (FInvert StepsSinceLastCatch) (Dot
           VSightCheck (SwapXY (ReplaceY FEphemeral(0.39355955085149197)
           VSightCheck)))) (Sin FEphemeral(0.43437104793302084)))
           (- (FInvert StepsSinceLastCatch) (Dot VSightCheck (SwapXY
               (ReplaceY FEphemeral(0.391738413114921) VSightCheck)))))))
      (- (MinXY (- (ReplaceX (VectorLength (VBranch
           HittingWall PredRotation VGet)) (VBranch (CSet
           CSightCheck) (Rotate FEphemeral(0.43437104793302084)
           VSightCheck) (VInvert PredRotation))) (Normalize
           (Rotate (VectorLength VSightCheck) PredRotation))))
           (- (FInvert StepsSinceLastCatch) (MinXY (- (ReplaceX (-
               (* (FInvert StepsSinceLastCatch) FEphemeral(0.9388))
               (VectorLength PredLocation)) (ReplaceX (BreakX (Rotate
               StepsSinceLastCatch VSightCheck)) (Normalize VSightCheck)))
               VEphemeral(0.25338263178802295,0.0040478457160352 86)))))
      (MinXY (- (Scale (FBranch (CSet (> (FInvert
           (BreakY VEphemeral(0.8184281590015943,0.3804437640803845)))
           (Dot VSightCheck VProximityCheck))) (- (FInvert
           (Dot VGet VSightCheck)) (- (FInvert StepsSinceLastCatch)
           (Dot VSightCheck VSightCheck))) (MinXY (-
           (SwapXY PredRotation) VEphemeral(0.2531,3.5770E-4))))
           (ReplaceX (FInvert (* (VectorLength VSightCheck)
               FEphemeral(0.9336392429394589))) (VBranch (CSet
               CSightCheck) (Normalize (Rotate (VectorLength
               VGet) (SwapXY (ReplaceY FEphemeral(0.39976556374102745)
               VSightCheck)))) (VBranch HittingWall PredRotation
               VGet)))) VEphemeral(0.25312640630028027,3.5770691168901525E-4))))
      (Dot VSightCheck VSightCheck))) (MinXY (VBranch
   HittingWall PredRotation (Scale (FBranch (CSet
   (> FEphemeral(0.43437104793302084) (- (FInvert StepsSinceLastCatch)
       (Dot VSightCheck VSightCheck)))) (- StepsSinceLastCatch
   (- (Dot VGet (Normalize (SwapXY (ReplaceY FEphemeral(0.3881134290499777)
       VSightCheck)))) (- (Dot VSightCheck VSightCheck)
       (- (FInvert (VectorLength VProximityCheck))
           (Dot VSightCheck VSightCheck))))) (MinXY
   (- (ReplaceX (FInvert (Dot VGet (Normalize (SwapXY
       (ReplaceY (VectorLength VEphemeral(0.8228,0.2819))
           VSightCheck))))) (VBranch (CSet CSightCheck)
       (Rotate StepsSinceLastCatch VSightCheck) (VBranch HittingWall
       PredRotation VGet))) VEphemeral(0.25312640630028027,3.5770E-4))))
   (+ (SwapXY (Rotate StepsSinceLastCatch VSightCheck))
       (VSet (Scale FEphemeral(0.8007832035440299) VProximityCheck)))))))
   (+ (SwapXY (Rotate StepsSinceLastCatch VSightCheck))
       (VSet VEphemeral(0.35333087982965106,0.13453288258312113)))))
```

Figure E.3: Full Genetic Program - KNNA 50F/50D - Part 2

The second half of the code for the best individual of run 01 for KNNA 50F/50D. Corresponds to the individual in run 01 in Figure D.8