Evolving Predator Swarm Intelligence

Fredrik Lillejordet

Master of Science in Computer Science
Submission date: June 2015
Supervisor: Keith Downing, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Fredrik Thorbjørnsen Lillejordet

Evolving Predator Swarm Intelligence

Master thesis
Supervisor: Keith Downing, IDI
June 2015

Artificial Intelligence Group
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering

Abstract

This master thesis investigates the evolution of predator swarm intelligence in predator-prey relationships. It aims to prove that we can evolve more efficient behavior than a set of simple rules in simulated environments. The thesis concludes that this can be achieved by combining vision models, genetic algorithms and artificial neural networks. The combination is well suited for swarming problems; where size, combinations and constant change makes it very complex to design simple rules. Our world is full of problems like this, where swarm intelligence and swarm robotics can provide effective solutions. Experiments investigate if and how performance is affected by changes in group size or environmental factors such as wall sight and noise. If robots are supposed to live in the real world, they need to be able to learn in realistic simulated environments. An observational study experiment is done in addition to simulated experiments; it shows that the evolved swarms clearly display intelligent swarming behavior.

Preface

The thesis is written at the Department of Computer and Information Science (IDI) at NTNU during the spring semester of 2015. The project is built upon ideas that emerged from a preliminary master project conducted at NTNU. The concepts for master projects are provided to students by the IDI faculty and professors at NTNU. Students are then free to explore the concepts further, and pursue related ideas. Projects form a research foundation, as a starting point for the master thesis to rise from. My supervisor gave me my initial concept, which was to investigate the emergence of global structure through local interaction in swarm intelligence. The results from the preliminary project created the seed for this master thesis.

Foremost, I would like to acknowledge professor Keith Downing, for giving me the opportunity to take on this project, and conduct research in my field of choice. I am grateful for all advise and support and he have given me, and for the freedom to pursue my own ideas. He has provided me with important insight and guidance. But most importantly, he has given me inspiration and motivation to explore new problems and possible solutions.

I would also like to thank my dear girlfriend Linn-Kjersti Sundar, for constant motivation and encouragement to pursue my dreams. Finally, I would thank my parents, family and friends for all your support.

Fredrik Thorbjørnsen Lillejordet Trondheim, June 14, 2015

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Spatial relocation relative to multiple moving objects has always been a challenging task, especially if the objects are moving in different directions, away or in a seemingly random way. The difficulty of this task is in the continuous need for successful predication, predication based on constantly changing and possibly information. Even if objects properties such as location, type and velocity can be fully observed, the task is problematic as these properties could alter or be seen differently from your next point of view. One wrong move could result in loosing the sight of the objects and an unaccomplished task as a whole. Reliable prediction of safe and efficient movement are desired solutions in tasks like this, solutions that could prevent failure and yet able to reach the objects within the desired time limit. Nonlinear movement of one object is difficult to enough to predict alone, even more so if multiple objects are involved. If all possible move scenarios are known, one can certainly engineer basic simple rules for behavior, that can handle those optimal every time. But, the same rules can prove totally useless if only new scenarios are added, or if existing ones change.

Tasks like this are visible throughout modern society, yet no perfect solution or approach exists because the problems are too different, large or constantly changing; making them to complex for any universal equation. Clever mathematics and graph theory can provide solutions when finding the shortest routes between objects, but they can easily be exhausted or incorrect if objects move unpredictably or not observable. Omniscient knowledge is unfortunately not often the case in the real world tasks. Here, it's more often that variables are unknown, seems insignificant or is too complex to predict. The reality needs approaches that can take intelligent decisions based on more information, learn, and adapt based on feedback. We need approaches to identify solutions we can't find.

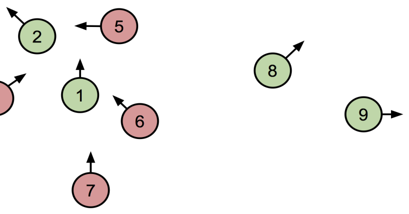

Figure 1.1: Spatial relocation relative to multiple moving objects

A typical problem with multiple moving objects: How can the red capture green most efficient? An obvious approach is to let red follow the closest green until they have captured all. Another possibility is to write a big set of behavioral rules stating what red shall do in all possible situations. A more interesting way, and the topic of this thesis is to just let red observe their surroundings and learn how to capture green themselfs.

Nature's evolved predators face similar challenges every day and their success in predicating prey movements decides who gets to eat and who don't. The predator-prey relationship is classical survival and comes from the core of evolution, it is straightforward and so feasible to abstract that it should inspire us to engineer solutions for similar real life challenges. Challenges where relocating resources in environments with changing information is highly dependent of good predication to ensure success. Our need for such systems is endless and can be employed in many areas of society, some easier imaginable than other. Hostile military analogies for predatorily systems quick comes to mind, where offensive systems and resources constantly move towards the areas that need them the most, whether they are patrolling, experience threat or engaging multiple enemies unprovoked. Other system exists for humanitarian reasons, such as fully autonomous location of persons, equipment or vessels in desolate places, distribution of aid packages, coordinating medical personnel or safely routing traffic. Whether we use the approach for military purposes or not can be argued irrelevant, it's the approach itself which are interesting so far, and which deserve our focus. The ethical discussion related to creating technology that can be used by either hostile purposes or by a futuristic artificial super-intelligence exceeds this project, and will not receive much attention. The attention will be directed out how we can let us inspired by hostile behavior in nature, as they display interesting spatial relocation of groups. A key factor of success with nature's predators is that they often hunt in groups or swarms, leveraging their power in numbers to find, hunt and eat their prey. And by doing so employing evolved strategies to

outsmart preys co-evolved escape strategies. This could be replicated when creating multiple spatially moving autonomous agents, by constructing our nature inspired intelligent predator swarms using simulated evolution to find the most effective strategies in every scenario.
Artificial Intelligence (AI) in Computer Science is the study of how we could construct intelligent machines. Because intelligence is a characteristic often thought to be reserved for higher functioning living things, the field is often met with skepticism, saying that machines are not alive and therefor cannot be intelligent. Optimists claim that we create systems that observably take smart decisions, behave intelligent could therefor be considered intelligent. Naysayer's claims that the machines are only advanced statistical algorithms, performing calculations to determine actions within a range of possibilities given by its human creator. Regardless of standpoint in this debate, it is clear that the creativity in a solution derived by a computer is often the main subject of this discussion, and researchers should therefor seek to expand possible actions for the machine to choose from. Thereby will letting AI create solutions we researchers can't find ourselves provide a greater form of creativity and possibly more intelligence in the eyes of the observer.
Swarm Intelligence (SI) is a subfield in AI that study the collective behavior that emerges when decentralized cooperative agents in a population infer actions based on what other nearby agents are doing, creating a self-organizing super organism. This emergence of self-organizing behavior is often observed in nature, and can be very attractive to replicate for because of its elegant simplicity, robustness and effectiveness. Nature's evolved swarms are simple in the way that each individual in a group is mostly governed by the same simple rules, rules evolved to ensure individual and group success. Groups can most often afford to lose a few individuals, making it a fairly robust bunch as few or no one is irreplaceable. Their size also helps secure effectiveness as each individual will contribute to common challenges and share success with the rest. This combination of features proves them effective as a group and interesting field of study. Nature's many insect colonies, immune systems, bacteria growth, bird flocks, fish schools and swarms forms important inspiration for further SI research, inspiration we could benefit from when creating AI multi-agent systems.

Engineering creative and effective swarm intelligence can be a challenging task because of its infinite complexity, but by combining inspiration from natures' predator-prey relationships, AI research and creativity; I will investigate how we can evolve efficient SI behavior.

1.1 Task description

The task of this project is to engineer predator swarm intelligence through artificial evolution that perform better than simple rules in a controlled simulated environment. The ultimate utopian task should be to implement SI in real world swarm robotics, which can benefit humanity, but this challenged would be to great for this project. The reasearch focuses on investigating evolved predator-prey swarm intelligence in a simulated environment. Predator agents should be able to observe other nearby predators, prey and walls within a given vision range, these observations should be the considered as local interactions and create a global structure of predators. Spatial movement of this global structure will be self-organizing, fully autonomous and considered as a strategy for hunting prey. How each predator react to the sum of current observation should be the product of a Artificial neural networks (ANNs), evolved by many generations of predator ancestors and Evolutionary algorithms (EAs) called Genetic algorithms (GA). The predator strategies should only be the result of local interactions, and therefor not require any previous planning or memory. This simplification enables the project to use a simple ANNs called Feedforward networks as decision-making system.

1.2 Goals and Research Questions

Goal *Gain understanding about how evolved predator swarm intelligence can be used instead of simple rules.*

Traditionally researchers have constructed sets of simple rules to control SI behavior, deciding how agents should behave in different scenarios and states. These rules are commonly used to design flocking behavior or create stigmergy. The goal of this project is to investigate how predators agents can employ evolve GA-ANN SI behavior effectively in different predator-prey scenarios, and specifically if GA-ANNs is an attractive alternative to simple rules.

Research question 1: *Can GA-ANNs be proved to sucessfully evolve SI behavior?*

The question will investigate if and how an GA-ANN implementation can evolve SI behavior, and if this can be proven.

Research question 2: *Will GA-ANNs benefit from manual manipulation of genotypes?*

Genetic manipulation and enhancement is important if preferable genotype manipulations can be identified and exploited, without loosing exploration abilities. I will research this by manipulating genotypes and comparing with normal ones.

Research question 3: *Does group size scaling of affect performance of previously evolved agents?*

The research question seeks to investigate if the efficiency of a SI evolved with a constant number of agents will change if the population size changes. The strategies have evolved because they performed efficient with that size, and it is therefor very possible that a change in population size will affected the results. The approach is very attractive if scalable without significant performance loss, because simulations are computationally expensive and this would allow researchers to use fewer resources when creating large population swarms.

Research question 4: *Will the visibility of walls or environmental noise affected performance?*

I will research if the visibility of walls surrounding the closed environment affects how predators hunt prey and evolve. It is natural to believe that predators would create strategies and tactics that use the walls to their benefit when surrounding prey, as they would know when prey limited escape options. This however is not guaranteed to be true in simulated evolved environment. Research should be done with the ultimate goal that it could be used in the real world and benefit humanity, and unfortunately the real world is a noisy place very unlike simulated environments. It is therefor important to research if the intelligence we create is sensitive to noise, in this case visual noise in form of small random numbers fed into the predators ANN in addition to the normal visual observations.

Research question 5: *Will the evolved agents clearly display SI behavior*

There exist many behavior patterns of behavior can be classified as swarm behavior. Typical behaviors such as flocking, herding, and stigmergy are common to observe in simulated swarms. I will research what is visible in the evolved swarms by observation.

1.3 Research method

The research is carried out in a simulated environment where everything can be controlled and measured. It is done so by many reasons, the most important reason is that the work relies on using genetic algorithms to evolve swarm intelligence behavior in predator-prey scenarios. The GA-ANN approach needs a large number of runs to simulation the natural selection necessary to produce good behaviors; this would not have been feasible with physical robots. Secondly, the simulated environment provides the possibility to manipulate all aspects of the observable environment such as agent positions, weights, noise and speed instantly; the manipulations enables more freedom to pursue interesting scenarios. The third and final reasons is that engineering multiple physical robots would have been a tedious, expensive and time consuming task, which could quickly have led to false- or no results in case of hardware engineering difficulties or equipment malfunction.

1.4 Thesis structure

The thesis is structured as follows: Chapter 2 presents the background information and why it motivates further investigations. In Chapter 3, I will go through the implementation I have created to accomplish my research. Chapter 4 explains how I have planned and conducted experiments a discussions of the limitations of the implementation. Finally in Chapter 5, I will conclude on the results produced by this research, review research questions and suggest what needs more work in the future.

Chapter 2

Background Theory and Motivation

The background of this project is to evolve predator swarm intelligence in predator–prey relationships using evolutionary algorithms and artificial neural networks. The subject is inspired by nature, but neither predator nor prey is supposed to imitate a specific species' behavior, the idea is rather to investigate if predator swarming strategies will evolve effective and creative strategies if given the freedom to adapt.

2.1 Background Theory

2.1.1 Predator-prey

The relationship is basic and found throughout nature: A predator is an organism that hunt and eats another organism to survive; prey is the organism that the predator eats[20]. The prey concept is not limited to animals, it can be any living organism: such as an apple or plankton. Survivors of this relationship will pass their genes one to the next generation, which will evolve further and thereby increasing its chances to balance their offspring's future in their favor. The evolved generations can feature new or altered attributes, which again would be passed on if proven useful. These attributes can be anything biologically possible, but usually vary from improved eyesight, stamina, smell, hearing and camouflage. It's not only improvements unfortunately, some negative abilities are will also be kept, this occurs because these individuals were more successful than their opponents in despite of these abilities. The evolution of useful attributes in predators is a fascinating cycle, but only a necessary process for the primary objective: the hunt for prey.

Hunting strategies can possibly be what best defines a predator; each usually involves some typical phases. These phases can be described differently but the most common is how they find, catch and kill their prey. They can often involve sub phases to better describe the series of events inside. Encircling, herding, separating are examples of events that could occur in catching phases where predators use these to control the spatial movement of prey to their benefit. An example of similar herding can be seen in nature's spinner dolphins:

Benoit-Bird [4] show that highly coordinated groups of foraging Spinner dolphins use herding to increase prey densities up to 200 times normal density. When the aggregated prey is dense enough, individual dolphins take turn eating, while the rest herd and maintain prey density. The coordinated dolphin strategy is deadly and though to be more effective than individual hunting, even though individuals use a lot of energy to help the rest of the group.

A successful hunt will sooner or later result in the final phase: the killing phase. It can involve heavy direct attacks, gradually wounding or a combination like paralyzing venom and later strangulation. Some strategies employ a combination of making the prey first suffer from fatigue or an excess of metabolic heat, followed by a finishing strike. The last strategy is usually called persistence hunting are believed [7] to be one of hominid's oldest forms of hunting. In contradiction to some four-legged animals - hominids have evolved the ability to regulate body temperature though sweating. This enabled hominids to pursue faster game until it were hot and had to lower its pace to cool down, making it significant easier to kill. I consider persistence hunting a killing phase because of the strategic weakening of prey, but it can also be considered a catching phase. Because of ambiguous examples like this, hunt strategies should be considered as a whole when trying to evolve artificially in simulated environments. The amount of work would be too time consuming if it were to engineer them separately and combine them successfully, extending the scope of this project. This extensive approach would have been interesting and should be the subject of further work. This project will consider predators in the broad conceptual way, not limited to specific animal specie. It's popular to create AI that focus on mimicking a specific predator-prey behavior or applying evolutionary pressure that can yield other alternative solutions to an existing behavior. These are some impressive SI implementations of predator-prey relationships done by researchers:

Olson et al. [22] have co-evolved spatially moving predators-prey relationships with the idea that: Swarm behaviors among prey have evolved flocking to confuse predator vision and thereby increase the chance for survival. Olson et al. [22] prove that prey successfully evolved strategies that used this approach, supporting the idea. Very similar approaches can be seen in nature, flocks of Australian wild budgies employ it to confuse nearby predators. [33] PBS [23]

Wood and Ackland [38] created a individual-based model for evolving group formation driven by the evolutionary pressure of selfish individual predator avoidance, not only collective group survival. Genetic algorithms are used to simulate natural selection and further evolution. Their model evolves on individual basis, giving each of the groups individual similar characteristics, but with some individual differences. The competing individual differences result in selfish behavior and an effective group as a whole.

Ducatelle et al. [11] created a heterogeneous group of cooperative agents who forage for food. The implementation uses an approach called ant colony optimization algorithm, but as they are locating and foraging something to eat, I consider them predators-prey because of the flexible definition.

The examples above are creative, impressive and motivate further research. Even though the approach of evolving group behavior is not new, it's reasonable to believe it still can provide surprising and effective solutions to new problems, as it have done in the past. The solutions from evolved scenarios emerge from the evolutionary pressure created by system designers, crafted to best describe what is needed to solve an abstracted problem without knowing the solution. This work provides a simulated environment with the freedom for predators to evolve their own strategies. I seek to be surprised by the results it yields, preferably good results, but that is beside the point. Predator evolution is helped to understand what is good and bad by reinforced by rewarding prey kills and apply small penalties proportional to time spent. This should make predators' pattern analysis easier, as they know what is wanted. It may seem ambiguous to say that good results are besides the point while reinforcing wanted behavior. This dilemma can be difficult when looking to find something effective and unexpected: you don't know how looks like; only what it should do. It is a balance tradeoff how much reinforcement is necessary to create effective solutions, and yet remain creative.

Nature and previous research tends to display co-evolution [22], where both predator and prey evolves at the same time to compete with the opposite abilities, thereby accelerating need for evolution. This work will not employ that approach, but rather focus on pure predator evolution in predator-prey scenarios, letting the prey behave approximately the same every time when identical events occur.

This is done to creative agents that can adapt to their surroundings, because I consider the ultimate goal of future work would be to create intelligent machines that benefit humanity in the real world. Their real world counterpart or objective would not be changing, learning or evolving in the nearly the same pace, making simulated co-evolution less useful when only the predator is needed. A better approach could be to better capture how this counterpart behaves in real life and simulate realistic alternatives to that behavior. This simplification through isolated predator evolution will focus how predator agents could best adapt to a current given set of problems, instead of trying to learn from scenarios that never could happen again.

Real world tasks

The end goal of nature's predator-prey relationships is undoubtedly survival, eat or be eaten. This is a fact and cannot be disputed, but I purpose that artificial predator-prey strategies will be employed to benefit many other purposes as well, especially when swarms of agents is used. The cooperating strategies multiple hunting agents use to locate and move toward another objective(s) are valuable and can be used in near endless scenarios. The eat/kill phase can be substituted and simplified with more friendly uses that only states that the objective is accomplished or handled, and that the hunt for the next objective may begin. The objectives in artificial predator-prey AI should be interchangeable with other difficult rapidly changing need-based tasks such as:

- Distribution of aid packages *(vaccines, food, materiel)*
- Provide network access *(Satellite networks, drone graphs)*
- Locate missing people or objects in remote areas *(at sea and in mountains)*
- Maintain agriculture crops *(only provide water and nutrients where needed)*
- Locate missing people or objects in remote areas *(at sea and in mountains)*
- Emergency response *(organize traffic autonomous to prioritize emergency personnel)*
- Autonomous redistribution of time expiring resources *(eatables and goods)*

It is easy to imagine real world tasks for smart artificial predator-prey intelligence if the objective changes, because the relationship is essentially based on effective spatial relocation of resources relative to need-based objectives. The evolutionary pressure of survival in these relationships demands constant adaptation of more intelligent behavior. This motivates the use of predator-prey when investigating swarming behavior.

2.1.2 Artificial intelligence

Artificial Intelligence (AI) is as mention in the introduction a subfield in Computer Science that focuses on construct machines or systems that can display what is commonly called intelligent behavior[27]. What Intelligent behavior is can be interpret differently: it can be behavior that chose the same decision a smart organism would choose, a better decision, a worse decision, or who chose no decision at all. The interpretation and definition of intelligence is subjective, it's not be the subject of this work and will not be addressed directly, as it would far exceed the scope of this project. A more interesting perspective is to assume that we are uninterested in defining true intelligence, only in creating systems that appear intelligent to unknowing observers.

'Whether a system is intelligent or not is irrelevant for the observer of such systems. - Shapiro [27]'

Focusing our research effort in creating AI systems can provide observable intelligent behavior and decision making to areas where intelligent behavior is needed. For a computer to display such behavior it would need three components: information about a problem, a system for making decisions based on that information and a way of expressing the chosen decision to users of the computer. The input and output are easy: Input can be given with sensors or buttons and output displayed with audio or visual signals. The difficulty is in creating an intelligent decision-making system, an AI. The AI needs to decide which signal it should output based on inputs and rules given provided the system designer, this is easy in simple cases when the result signal can be calculated, but quickly becomes impossible to calculate as input mass and combination grows. When this happens, designers have three options:

- *Accept that growing input and uncertainty will reduce displayed intelligence*
- *Create additional simple rules to cover more scenarios*
- *Provide the decision-system with the ability to learn from new scenarios*

The most difficult and yet most attractive option here is to enable learning abilities, it is also a common criteria when characteristic of intelligence [27]. Learning abilities can possible be the most important and central subfield in AI and Machine Learning because of its need and importance it's approached in many ways. The most popular is undoubtedly Unsupervised learning, Supervised learning and Reinforcement Learning [31]. Unsupervised learning tries to learn how to find a hidden structure or pattern in the stream of input information, by comparing input to previously given examples sets, without knowing what the examples are meant to represent. The process can be challenging and time consuming, as it does not know what the input or examples are, and because the system will not give any positive or negative response to correct or wrong answers. Supervised learning is similar to unsupervised learning, but the system here know what examples mean, and gets a suitable response to correct or wrong answers. This makes classification of input information easier and more predictable, but may limit the possibilities because the system will have more difficulty to recognize an input where no example sets exist. Reinforcement learning is used when the systems is rewarded if wanted behavior is displayed, and punished if unwanted behavior happens. Over time, this will help the system learn what is good and what is bad. This approach is widely used and can easily be implemented in all scenarios where the designer know what is good and bad behavior, by reward and punishment. Reinforced learning usually doesn't know what perfect input is, because it hasn't been shown examples, as supervised learning has. Because if its similar nature, Reinforcement learning will be fitting learning approach to use when engineering predators intelligence.

2.1.3 Swarm intelligence

Swarm Intelligence (SI) is AI for autonomous multi-agent systems that display swarm-like behavior. The term itself was first used by Beni and Wang [3] to describe how a finite number of autonomous cellular robot agents could accomplish a global task, only by using local communication. The agents would have low complexity, cooperate efficiently and designed to be highly adaptive to their surroundings. The number of individuals in a employed would ensure redundancy and reliability; affording to lose a few robots and still work sufficiently, providing graceful performance degradation [30]. The general idea was that combined groups could solve tasks no single robot could do on its own. Over time the concept has grown and more ideas have been in incorporated in SI, but the main idea of global structure formed by local interaction and cooperation remains in center. SI is possibly best known for solving various optimization problems such as Particle swarm optimization (PSO) and Ant colony optimization (ACO), these approaches are impressive but have totally different use cases than this project because they seek to solve static problems that don't change often. PSO [15] are used to find a global maximum in problems where we are dealing with the combination of n-different parameters represented by an n-dimensional solution space, each represented by a particle that is moving around with changing velocity. The combination of all particles in the n-space will represent the solution and hopefully a global max to the optimization problem. ACO [10] is a optimization technique inspired by how ants uses local interaction through pheromone distribution to help their peers find their way. It uses a similar probabilistic approach to explore a parameter space of all possible solutions to hopefully locate the best solution: An optimal path through a graph. This project does not use PSO or ACO, but rather inspiration from predator-prey scenarios and a hunt optimization. Even though effective swarm robotics is a strong motivation for further research, its not longer the only appliance for SI, it have been successfully been used in other areas: Such as simulated passenger crowding using ACO, to prove improve efficiency and experiment with different boarding strategies on airplanes [19]. This project will investigate relevant subfields pragmatically; it's not an attempt to cover the entire field of SI.

Self-organization and Stigmergy

Self-organizing [30] behavior is used when distributed agents cooperate by using local inter-agent communication. It could consist of anything perceivable by other agents: Direct or indirect audio, visual or physical signaling. Even observable spatial positions are enough to be called communications. Because all agents behave relative to what the rest is doing, each event could trigger an endless chain of events, hopefully establishing a favorable emergent functionality. This process where a chain of events occur based on indirect communications is also called Stigmergy[2]. It is inspired by how nature's social insects use indirect communications to from efficient collaborations among simple individuals; a typical example is how ants are leaving pheromone traces to indirect communicate a good route through a spatial graph. [37] [19] Self-organization and Stigmergy is attractive because the autonomous agents don't need to rely on external commands from outside the group; this makes them useful in desolate and inaccessible remote places like at sea, mountains and other planets.

Flocking and Herding

Flocking is the aggregated motion of multiple nearby agents or boids, when moving in the similar directions influenced by each other [24] [18] . The behaviors that lead to researchers creating simulated flocking by using a combination of collisions avoidance, velocity matching and flock centering [24]. European starling murmuration is possibly the most common example of flocking behavior in nature, the starlings can gather in flocks of hundreds thousands of individuals and fly together without a leader [22]. Its noted by Reynolds [24] that natural flocking instincts seem to be sharpened by predators, the evolution of prey flocking behavior have been observed in a simulated environment by [22]. This warrants the need for more effective predator intelligence when facing prey swarms.

Herding Banerjee and Banerjee [1] behavior occurs when agents with intent can control how others are behaving. While flocking is an aggregated motion where there may not be a specific goal, herding other agents is done with clear intent and goal by controlling the behavior of others, instead of letting them use their own information to make a decision. Both behavioral patterns are useful in SI, and can each contribute with their features are needed. It could be interesting with a combination in predator-prey SI scenarios: Where predators could flock around searching until prey is located and then herd the group until they are contained enough to begin the eating phase, this is done by the spinner dolphins described in predator-prey [4]. Another interesting example is how customer herding can be an effective marketing tool in supermarkets [36]. Owners can track and let customers herd each other by strategic positioning of merchandise, finding the most profitable combination of positioning. Similar approaches is done on a individual level by letting goods like chocolate and sweets by located near the exits, this approaches uses all customers as a herd and is much more interesting.

Robustness and Redundancy

Robustness and redundancy should in focus when developing swarm behavior. The behavior needs to be robust enough to solve tasks in different settings, even if the task and environment is perceived differently or reactions differ by the individual agents in the group. Multiple agent's cooperative who can replace each others if some fail provide valuable redundancy, this increases the groups robustness indirectly. More robustness could also be archived by reducing agent complexity, the tradeoff is that this could also reduce functionality. This reduction in functionality would reduce production costs as fewer components are needed, required debugging costs would also scale with the reduced complexity. This makes low complexity very attractive as long as the group can accomplish the given task, this surfaces another problem: How we can predict if our engineered swarm is robust enough to accomplish tasks without actually testing it? The short answer is: We can't. The long answer is: We could create a mock up simulated environment that replicates the real world to the best of our knowledge and test in similar real world conditions: like testing Mars Rovers in nearby deserts and simulate lower gravity.

The simulated environment would need to be perceived as similar as possible for sensors and actuators to work correctly, and even if we replicate everything perfectly things can change in the real environment. One solution is to engineer our system robust by using extra random noise that I could work OK under every condition. This would probably make it less sensitive and therefor lose some functionality, but if the real job is somewhere so remote (like another planet) that we cannot debug once deployed, that is an attractive solution. The real world is a noisy place; we should therefor create solutions that tolerate some errors and imperfections in observations.

Brooks and Flynn [6] argue for using small cheap cooperating autonomous robots in planetary exploration because they could among others reduce cost, implementation time and provide redundancy and reliability. Brooks [5] also mentions how robustness is important mobile robots: they need to work if some sensors fail or giving erroneous readings followed by self-calibration and quick recovery if it happened. He also argues that we are interested in robots that can survive months without human assistance, in dynamic complex environments. He's ideas are very important when thinking robustness and redundancy in SI.

Simple rules

In SI, it is common to create behavior by carefully construct a set of simple rules that decides how agents should do in every situation. In some cases this can create a relative realistic simulation of nature, like in algorithms that simulate the flocking behavior of birds in flight. These algorithms usually involve rules for collision avoidance (avoid nearby agents), velocity matching (match the other agents speed and direction) and flocking centering (stay close to other nearby agents) like mentioned earlier [24]. Simple rules are very effective in simple scenarios, but can fail catastrophically if the scenario is to complex, and situations where all or no rules apply. Example: If agents were not instructed to move and search unless they actually observed others, noise or obstacles could quickly cause total failure. Challenges where multiple options is attractive would force a decision response, and the outcome could be dramatically worse if the designer hadn't design the system to handle challenges like these.

2.1.4 Virtual environments

When experimenting with simulated virtual creatures or agents, a virtual environment for them to inhabit is needed. The virtual environment should be as realistic as practically possible because the usefulness of experiment results will scale with the level of realism included. Spears et al. [29] have noted how we could employ artificial physics and that they should live in a 3d-world, because we do, about artificial intelligences. Brooks [5] also state that the human world is three dimensional, therefor robots environments should be. The 3d arguments are simple and logical, and should therefor be taken in account when creating virtual environments. Artificial physics or 'physicomimetics' as Spears et al. [29] mentioned is also important, especially if the implementations is largely dependent of it to learn as what it should. Gravitation and centrifugal forces would be essential if creating spaceships. As with [14] work where virtual creatures learns to swim, reduced gravity is essential. Some another key aspects of real environments are noise and obstacles. Noise should simply be included because the real world is a noisy place; therefor the virtual environments should be. It could be a good idea to increase the amount of noise in simulations as performance increase, or evolve noise as well as solution [38]. Spatial limitations should always be included in virtual environments, because limitations such as obstacles, terrain and walls are everywhere in reality. All these elements should be considered when modeling virtual environments, the ones crucial for the specific scenarios success is necessary, but all should be considered as a missing essential modeling element can render the entire implementation useless.

2.1.5 Vision models

Agents should possess abilities to sense their surroundings [29]. Physical robots use different sensors to measure different physical phenomena, depending on what the goal is. Some sensors measure by using sound or laser, others may use video signal image analysis or even chemical signals [9] [2]. What and how surroundings are sensed depends on the implementations, different implementations have different needs and resources when it comes to cost, precision, robustness and resolution etc. It can be fatal for an implementation to employ the correct approach that captures and observe the surroundings correctly. The same idea goes for sensors in simulated agents, they to have to be appropriate for the intended usage. Simple simulated predator agents would need the ability to observe nearby prey and possibly also other predators and environment. This have previously been done in many different ways:

Olson et al. [22] created a directional 180-degree artificial retina model to represent predator and prey vision. The 180-degree retina was divided into twelve 15-degree vision zones used to separate where prey was located, giving the vision model a resolution. The models have the ability to distinct between predator and prey, giving individuals a idea of where it's safe. Prey vision is limited to 100 meters and predator 200 meters, giving the predator an edge when hunting as it can see prey before its seen. The implementation is created to investigate if prey can involve confusion strategies to escape predators by using flocking in front and hiding in its blind spot/dead angle.

Oboshi et al. [21] uses a two different vision models in the same system. One for predator fish and one for prey fish, both simpler than simpler [22]. The predator vision model uses two different radiuses two decide behavior by simple rules. Prey is captured if within the inner radius, and chases if within the second. If no prey is visible within the second radius, the predator will search the environment

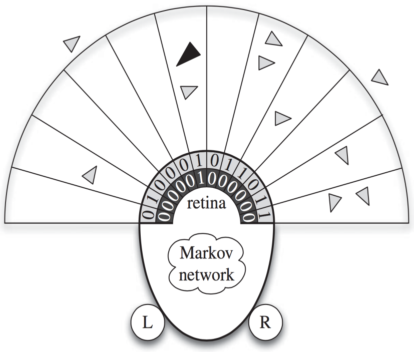

Figure 2.1: Predator-prey vision model images from Olson et al. [22]

for others. Prey vision model is a bit more complicated, it uses three radiuses to decide avoidance, parallel, attraction and searching. Similar to the flocking boids described in Reynolds [24]. Both models have a dead angle area behind agents, both much smaller than the 180-degree angle in Olson et al. [22].

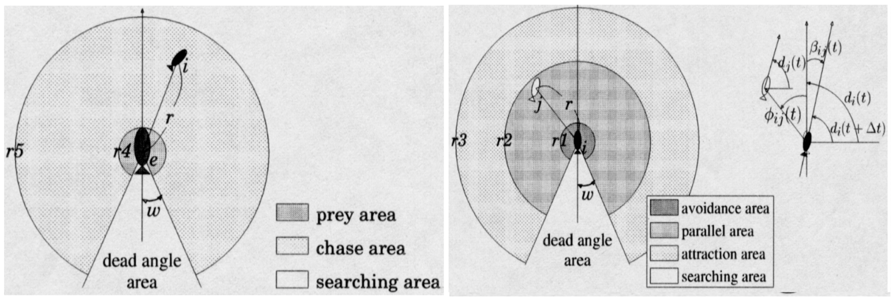

Figure 2.2: Predator-prey fish vision model images from Oboshi et al. [21]

Both implementations give agents a good impression of other nearby, but limits observations to a predefined range, so that searching becomes necessary. Neither of the agents observes environmental factors such as walls or obstacles. The implementations provide motivation for further development.

2.1.6 Evolutionary algorithms

Evolutionary Algorithms (EA) [34] is used to find solutions to complex problems by simulating evolution inspired by biological evolution. Although simulations are run in a virtual environment, the approach can be used to engineer any solutions in any form, physical and virtual. The virtual modeling of complex physical challenges can be difficult, but it's still a more feasible option than creating and testing millions of different physical prototypes. A well-known example of this simulated approach is NASA's evolved radio antenna [17].

Simulated evolution includes environments and processes like reproduction, mutation, crossover and survival of the fittest through natural selection. Individuals in the evolution are referred to as candidate solutions, as they are candidates for the final solutions that would emerge at the end of each evolution. The candidate solutions are represented in the evolutions by their own individual unique phenotype encoded from their genotype, which they got at 'birth'. Genotypes can be represented by large sets of simple atomic values such as bit arrays. Phenotypes are usually fewer in numbers but a little more complex and granular, as an example can they be in the range 0-255, or -1.0 to 1.0. How phenotypes are encoded by genotypes can also different in every system and done in many ways, both direct and indirect. A common indirect approach is to use sub-ranges from a bit array genotype to represent one granular phenotype number, by treating the genotypes as binary numbers and the phenotypes as decimal. This could make small changes in genotype represent big changes in phenotype and opposite, thereby increasing the diversity of what could happen. Another simpler solution is to use direct encoding where there is identical encoding in both genotype and phenotype, thereby removing the need for one and reducing the computations needed to run simulations. Whichever encoding seems most fitting in the current scenario is up to best practice and the system designer and to decide, because the important thing is that it works, not that is fancy.

The EA approach is to start with a generation of simple or random seeded candidate solutions, run simulations and perform individual fitness evaluation based on their performance in the simulation. After fitness evaluation we should keep and select those who performed good and let them reproduce and evolve by using crossover and mutations, at last select some of the new children candidate solutions and let them combine with their parents: creating a new generation of possible candidate solution. Lastly, repeat the process until an either a perfect candidate solution is found or we run out of resources or time. The best candidate solutions will now be an evolved solution, hopefully proven to perform well in the assigned task.

Crossover and mutation algorithm schemes alter from system to system, and can be the source of debate. It is common to use a predefined set constant percentages ranges and a pseudorandom number generator to change the value of genotype indices. Every time a new candidate solution is 'born', the agent would create a blank genotype and inherit each genotype index value from either the mother or father candidate solutions, from whom will be decided by creating a random number and copy from one parent if its within the given constant range, and copy from the other if outside. Mutations are similar; it will loop through the entire inherited genotype, creating a random number and perform mutation on that index if the random number was within the constant range. Mutations could be any altercations to the initial value, such as a bit- flip (1 becomes 0 and 0 becomes 1) or adding or subtracting small values (0.01). Different systems usually want to promote evolution process through focusing on different schemes: some focus on strong crossover processes (EA) and some focus on mutations (GA). A possibility is to let the EA itself control the process by evolve constant percentages or how the schemes work. Spears [28]

The designer of the system chose which and how different selection algorithms and schemes is used and performs. It depends on how often it's wanted to trim away candidates from the group, and which candidates should be removed. It's typical for EAs to mimic nature and have individual selections at big moments in life: who find a partner in life, which who children survive, and how grows up stronger than their peers. The EAs equal to this would be: Parent selections, children selections and adult selections. Other popular selection approaches is: elitism (where only the best is kept), generation replacement (where only the children survives) or mixes where selections uses combinations to provide the wanted diversity.

It is important to evolve solutions that could perform acceptable under every condition, not just perfect in some. This can often be accomplished by increasing the candidate solutions diversity by letting solutions try different similar task with the possible range and also keep some mediocre candidates and genes through crossovers. And by using a fitness function that not only rewards perfect behavior, but also rewards minor accomplishments. All done in hope of keeping candidate solutions that solves task both creative and effective and avoid very over-fitted ones, as they tend to be very good at some problems and terrible at other.

The most common EA is called Genetic Algorithms (GA) [12] [16] [22]. It is typically used to create virtual sets of characters or numbers directly represented by the best solutions phenotype, evolved by the mentioned processes. These sets can be used for many things: such as properties in complex system, letters in handwriting recognition or in this case: weights in artificial neural networks.

One of the most famous recent examples of evolved intelligence was created by [14] in 1994, Sims work is displaying that virtual sea creatures can evolve to accomplish simple tasks like swimming and capturing objects, if given clear tasks and freedom. The creatures could either evolve on their own, or co-evolve competing with others.

2.1.7 Artificial neural networks

Artificial neural networks (ANNs) [12] [39] is statistical learning models used in AI and cognitive sciences to take decisions based on sensory inputs in complex tasks. ANNs are inspired by the central nervous systems in animals, and is very simple abstractions by comparison. The use of ANNs in AI is not intended to replications of the complex human brain; it's only to provide systems with the ability to take good decisions based on a large amount of information where engineering simple rules may be to complex. Nevertheless, ideas of how networks of dendrites (connections) connecting neurons (nodes), and how some chemical combinations inside neurons can trigger and fire (activation function) small electrical signals have enabled researchers to create impressive implementations. A shared idea how ANNs work is that sensory signals is feed into a set of input nodes, signals then propagate further into a networked graph of connected nodes combining and crossing signals, until a signal reach an output nodes, triggering a action represented by that output node.

The system makes decisions on which action to perform based on the current sensory input based and how the network of nodes is. The strength and direction of each node-to-node connection and network topology is usually decided by a set of adaptive numeral weights and an activation function. The combination of these will usually decide the performance of the ANN within its own capacity of success. The activation function is static and decides how a signal has to look like for it to be fired and propagated further in the net. The number of weights representing nodes and/or connections will scale the ANN complexity and there provide possibly better solutions, but also increase the difficulty in finding a solution significantly. Good performance are largely dependent on finding good weights, and that can be a cumbersome process if done manually, this work will therefor employ EA to evolve weights through simulated natural selections.

The usage of ANNs is widespread in computer vision, speech and handwriting recognition and self-driving robotics. Different use-cases, needs and complexities have led to many different implementations of ANNs with, each having different features, strengths and weaknesses. The least complex solution that solves a given task will often be the best solution, as is reduces the cost of engineering and debugging time significantly. This especially applies to ANNs, because the complex can be scaled infinitely. Therefore we should try the least complicated thing that could possible work, and only extend if needed. This project will use a relative simple ANN modeling called Feedforward Network for reasons mentioned in implementation.

Feedforward neural network

Feedforward neural networks (FNNs) [35] are relative simple ANNs where signals only travel in one direction through connections across the network. Signals can multiply and propagate through several connections, but never travel the opposite way. Unlike more complex bidirectional networks like recurrent neural networks (RNNs) [25]. RNNs signal can travel both ways or signal can be retained until the next round of sensory input. FNNs was the first and simplest form of ANNs, they usually consist of an input layer, one or more hidden layers and an output layer. Each layer can contain any number of nodes, but the number of nodes in the input layer will be the same as the amount of sensory inputs provided. Similarly, the number of nodes in the output layer(s) will be the number of different actions the systems should be able to decide from. The number of hidden layer(s) and number of hidden layer nodes in FNNs can be anything above one, all depending of how complex the process of deciding action should be. If no hidden layers were used, only input and output layer: the ANN would be considered a Perceptron [8]. Example: If the FNN would receive a yes and/or maybe answer from 3 people and provide a yes/maybe action the number of input nodes would be 6 and output nodes 2. The number of hidden layers and nodes is not determined by input or output, but should be at least one.

A early and important example of evolved 'neural networks' was created and presented by Selfridge [26]. He didn't called it a neural network, but rather a Pandemonium, which is a box of weighted connected cognitive demons nodes and a decision demon. The demons were to receive information from a data or image, this signal would propagate though the weighted demon graph, and each demon would yell its assigned unique letter to nearby demons and thereby give of a signal. The signal graph forming a hill-climbing problem and should result in a decision at the global maximum. Selfridge also want promote an evolutionary process by having weights and functions adapt by usefulness.

2.2 Motivation

The background research motivates further work; Reinforced learning using GA-ANN has great potential in creating better predator-prey swarm intelligence than creating sets of simple rules. I believe this because the approach could provide more creative strategies that have already been proved effective through evolution. In scenarios where large co-existing swarms of predators and prey would result in virtually infinite different states, the challenge of designing simple rules, handling all possible states effectively is near impossible in complex scenarios. It is a better-suited approach to let the system adapt itself, by letting it the AI experience and learn from all the given scenarios, and find a strategy that works good every time. Adaptive approaches will become necessary if AI should be used in the real and noisy world, human designed simple rules cannot successfully capture all possible combinations of events that may occur. The predatory nature is highly adaptive to surroundings; the underlying raw hostility has justified nature's need for evolution for millions of years, this makes predator-prey swarms' very inspiring and perfect pragmatic motivation when engineering evolved self-organizing swarm intelligence. The most common related research area is to investigate how prey can evolve to escape predators, not how predators can hunt prey. I find this interesting because I consider the latter more suitable for real world challenges. This motivates further research focus on increasing predator behavior performance while prey behavior remains the same, as I consider isolated predator evolution most relevant and necessary when creating real world solutions.

A inspirational quote by Oliver Selfridge have stood out among all research I have read, its describing his early idea of 'evolved neural network swarms' – evolved Pandemonium crowds:

'It is perfectly reasonable to conceive of this taking place on a broader scale – and is fact it takes place almost inevitably. Therefor, instead of having but one Pandemonium we might have some crowd of them, all fairly similarly constructed, and employ natural selection on the crowd of them. Eliminate the relatively poor and encourage the rest to generate new machine in their own image. - Selfridge [26]'

Chapter 3

Implementation

This project has been researched using the implementation explained in this chapter. It's engineered specifically for this project, and portrays a general abstraction of nature's predator-prey relationships. It's capable of running entire hunt/escape simulations with any number of predators and preys in a wall-closed or torus environment. Simulated predator agents use GA evolved ANN for deciding movements, or simple rules for comparison, while prey agents only use simple rules to move. Observers can study pre-evolve ANNs simulations by using a third party visualization library called Three.js, visualizations is used to investigate if signs of typical swarm behavioral patterns is apparent.

3.1 JavaScript

The implementation model is built entirely in JavaScript programming language. It includes an evolution simulation, which can be run in a command-line using Node.js [13]. It also includes a visualized implementation that can employ evolved weights to study how the weights and settings correspond to an observable behavior. JavaScript have mainly been chosen for the following reasons:

- **Dynamic typing capable of handling of objects and adaptation flexible**
  Creating, storing, extending and returning objects are done often throughout the code. With dynamic typing, it is easy to adapt functionality if the existing objects change or if other information is needed to succeed

- **Command-line and Browser runtime compilation**
  Runtime compilation makes experimenting with new ideas agile and less

time-consuming. The reuse of code, functionality and language in both visualization and simulation makes development much more effective as code only has to be written once.

- **Prototypical inheritance modeling**
  Prototyping in JavaScript is an effective approach for models inherit others models. This way, core functionality only has to be located in one model, to be inherited and specified by others.

- **Third party libraries**
  JavaScript's supply of third party libraries is extensive, which makes it easy to find a fitting library if needed, such as Three.js for visualizations.

3.1.1 Three.js

Three.js [32] is popular JavaScript library that makes programming 3D visualizations trivial. It feature built-in support for creating box shapes, materials, colors and lighting conditions. Because of its wide popularity, it's easy to get support and see best-practice approaches. Unfortunately, it does not include a physics engine and object collision support, these have to be programmed or found elsewhere and combined with Three.js.

3.2 Agents

All agents, predators and preys share some functionally; this is located in agentController.js. The agentController contains many variables and helper functions, but most important is a **simple rules** algorithm called **huntOrEvade**. Its used for moving relative to the closest nearby enemy within a given vision range. Predators can use this algorithm to hunt, and predators can use it to evade, the use depends if the result of this function is negated or use directly. Current agent speed is multiplied with the function result so that predator and prey speed can be adjusted independently.

Even though they have edge detection and different speeds in different states, the rules only have two main objectives: 1: Escape the closest enemy if someone is visible, 1: move towards the center if nobody is visible. Simple rules **huntOrEvade** algorithm is computed as follows:

1. **FIND:** Find the closest enemy by comparing enemies distances
2. **MOVE:** Add move toward the center if no enemies are nearby.

3. **MOVE:** Add one move unit in both width and height direction relative the closest enemy
4. **MOVE:** Add one move one extra unit in one direction, if the other direction is not used
5. **MULTIPLY:** Multiply moves with the speed variable
6. **EXECUTE:** Exectue moves
7. **CHECK:** Check with torus or edge functions if agent is outside environment.
8. **WIGGLE:** Apply small random movements(wiggles) in addition to the executed moves.

The algorithm enables agents to execute moves relative to nearby enemy agents, but it does not handle friendly agent collisions, differ move relative to the actual distance to closest enemy or consider two or more enemies at similar distance, only the closest enemy. The algorithm is simple, but effective and can be used by both predator and prey. Each of their respective controllers inherits the agent controller and its functions.

3.2.1 Wiggles

In nature, millions of small biological events happens every second, events so small and insignificant we call them random. These events and processes ensure that nature keeps evolving and avoids stagnation. I will therefor include this in the implementation so that agents can benefit from small random spatial movement, called wiggles. The agents would perform extra random wiggles every time step, in addition to the movements decide by the ANN.

3.2.2 Prey

Prey inherit the agent controller, and use the simple rules algorithm huntOrEvade as a vision model with a vision range of 250 move units, and a speed of 1.5 units per time step. This enables them to observe and move away from predators that are a distance of 250 units away, and move towards the center if no predators is visible.

3.2.3 Predator

The predator agents can also use the simple rules algorithm, but they can also use a more creative and adaptive approach to hunt, the evolved neural networks

(ANNs). All predators are homogenous, meaning that the swarm can either use simple rules or ANNs, not both at the same time. Every agent has to use the same decision system. This makes other predator behaviors more predictable, as they all would take similar decisions, with just small differences depending on additional observations, noise and small wiggles. By observing each other, predators communicate indirectly though observable behavior, and thereby forming a global structure by local interactions, called Stigmergy.

The ANNs algorithm decision-making works in the following way:

1. **Impressions:** Nearby predator, prey and wall observations get computed into input signal impressions.

2. **Measurements:** The signals are measured, strength decreases approaching zero as distance increases until they are outside the vision radius.

3. **Inputs:** Input signals are sent into the ANNs, where they are combined as into input layer neurons.

4. **Propagation:** The input signals propagate through the input, hidden and output layer.Each signal passes through an activation function which decides further signal strength.

5. **Decision:** The ANN uses output layer signals to decide an action, reflecting which vision zone angle are most attractive to move towards.

6. **Position calculation:** Next x and y spatial values are calculated using cos, sin and provided angle and predator speed

7. **Edges:** Torus or edge functions checks and adjusts if the agents are outside the environment.

8. **Wiggles:** Wiggle function applies small random movements to the agent in addition to the actual move

The ANNs algorithm provide more information and a more complex decision making process. This can possibly create better solutions, but does also require much more computing resources (CPU, RAM). This will make the feasible maximum number of simulated evolution agents lower when using the ANNs algorithm, than simple rules algorithm. Because of this, it is attractive to evolve behavior using a small amount of agents that also would work if the amount scaled up.

3.2.4 Vision model

Predators observe can their surroundings with a vision model: a 360-degree artificial retina divided into twelve 30-degree vision zones, representing the resolution of the vision. Each zone gets three separate inputs to differentiate between predators, prey and walls. Multiple similar agent impressions in the same zone will be summed into one input signal. All 36 input signals will be weighted in the ANN. Olson et al. [22] used a similar approach when using evolved swarm intelligence to test if prey evolved behavior that could confuse nearby predators. Their model was a 180-degree 2D retina, capable of observing nearby prey only. The limitation posed by 180-degree vision enabled prey to benefit from predator blind spot. This work uses a vision model which is created to let predators get a full 360 impression of their environment, including predators, prey and walls relative to distance. The visual impressions of similar objects in a vision zone is combined into one input, so two prey in one vision zone only becomes one input signal. The vision impression resolution is half of what [22] used, but will still provide predators with great possibilities to evolve creative and effective strategies, using nearby predators and walls to their benefit in the hunt. The predators have a vision range of 300 units.

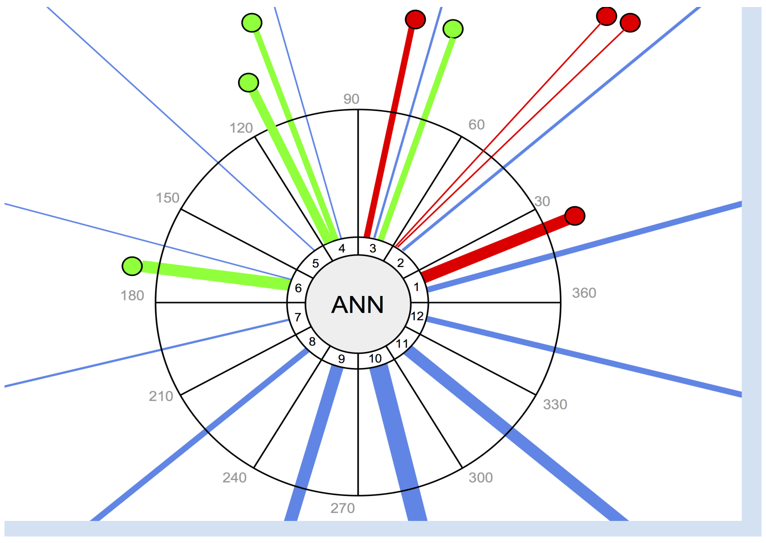

Figure 3.1: Vision model: The illustration show how an agent positioned in a corner employ the vision model to observe nearby predators, prey and walls.

*Vision zone values recieve a input of 0 if no agents are located that spesific vision zone. The model gather a total of 36 inputs, 12 zones * 3 object types = 36. A spherical 3D extension of this model would use 12 * 12 * 3 = 432 inputs, if the same resolution were used. Or 432 / 2 = 216 in a 180-degree vision model*

The environment does not provide any built-in functions to perform object observations, these is custom implemented. The necessary observations are to detect and measure distance to other agents and walls.

Agent detection

To detect agent observations inputs, both predator and prey agents are loop through every time step, investigating each individual agent. The absolute distance and angle to every agent is calculated by using their relative x and y positions, and used to calculate how strong the input signal should be, and in which vision zone it belongs. The input signal is normalized so it gives a value between 0 and 1. The code is located in the getInputData function in agentController.

Wall detection

Wall detection is somewhat more complicated because I as the system designer construct the walls; they only exist where I make them exist. The approach is to calculate distance to nearby walls at all vision zone angles is based on calculating the distance to line intersections, because where the wall should be, is information I control. The first challenge is to find out which wall; the vision zones angle points towards, secondly the intersecting point on that wall has to be found, and at last, the distance to the intersecting point needs to be calculated. The algorithm is designed to be as fast as possible by using trigonometric, because it has to be used every time step.

The code is located in the getAgentWallInputs function in predatorController and the algorithms is as follows:

1. Defining a radius **R** large enough to always be outside the environment
2. Defining the agents angle to all the environments corners **C** (northwest, northeast, southwest, southeast)
3. Defining a distant point **X** outside the environment by using the angle **A** in the center of the vision zone and **R**, and make an imaginary line of sight **L** to this point.

4. Check if the angle **A** is identical to any of the environment corners, if so: calculate the distance to the known corner location **C** and jump to 7, continue if not found.

5. Check which two corners **C** angles the angle **A** is within, these corners define which wall **W** line to use.

6. Calculate the distance to the point **P** where the line **L** and **W** intersect

7. Use the distance as input if it's within the defined vision range **V**

8. Repeat the algorithm on all 12 vision zones

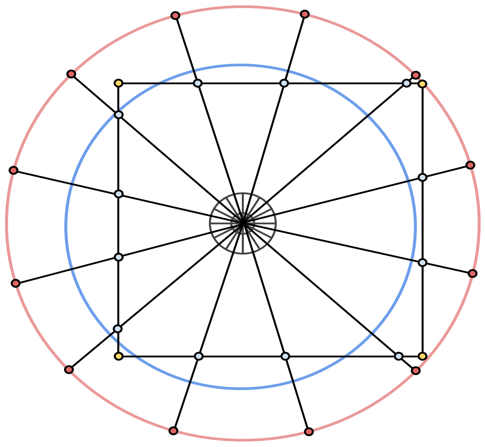

Figure 3.2: Vision model: Wall detection.

*The illustration show **R** as the outer red circle, yellow corners **C**, Red distant points **X**, black lines **L** from the agent with angle **A**, walls **W** between two yellow points. Blue interesection points **P** and vision range **V** as a blue circle.*

3.3 Environment

The agent environment is a rectangular box with the width and height of 500 move units. The environment is created in 3D with Three.js, but agents only use the x- and y-axis. It was intentionally designed to use the z-axis as well, but that would have increased the required calculations dramatically if the predator vision resolution would be kept as is the described vision model. The environment uses no form of artificial physics or object collisions, other than containing walls and floor. The walls are observable by predators, and be used to their advantage. The predator implementation has the ability to add random addition random noise to the sensory inputs, at all predator, prey and wall sightings. The noise is added to simulate errors in visual sensors and distance estimate assumptions in the real world.

3.3.1 Scenarios

The implementation includes some predefined initial start positions to be used in the environment; this is suitable make comparable simulations and reproducible results by using similar scenarios. The implementation also includes generative and random approaches to assign positions; the generative approaches distribute agents along a vertical or horizontal line and the random approach assign pseudorandom initial positions. The generative and random positions are used to avoid overfitting behavior, performing perfect in only some predefined positions. The idea is to let agents learn on a set of different scenarios, some predefined and generative to increase performance in known scenarios and some random to increase creativity in all possible scenarios. All scenarios are in the appendix, viewable as images.

3.4 Genetic Algorithms

Genetic algorithms (GA) are used to evolve solutions directly encoded into weights use by predator's artificial neural networks (ANNs). The best solution is selected from a population pool of possibly candidate solutions, evolved through many generations of selections. The selection process is to test the solutions performance in simulations and conduct fitness evaluations. The fitness decides which solutions reproduce new children solutions in their image by using crossover. The solutions are mutated to possibly produce new attractive behavior. The solutions that don't reproduce will be discarded; only the parent and children solutions will represent the next generation's population.

Each candidate solution individual represents a set of weights used to determine how a group of predators should decide actions by using the weights in their identical ANNs. Every predator in the group will therefor display same behavior in identical scenarios.
The GA doesn't use a typical indirect genotype to phenotype encoding, but rather a simple direct type called solution as genotype, without extra mapping. Solution is an array of numbers, each index represent a neuron weight in the ANNs. The weights can be anything in the range of -0.1 to 1.0; higher numbers will increases

the chance of neuron firing signal further.

3.4.1 Fitness function

Simulation performance is measured after each simulation by assigning a fitness score; the entire predator groups share its fitness score, as it's the groups' performance that's interesting in swarm intelligence. The individual fitness scores are aggregated and averaged if multiple world scenarios are simulated with the same weights. Using multiple different scenarios is a safeguard against overfitting the group behavior. The implemented fitness function consists of two factors, each theoretically capable of receiving a score of 50, together 100. The first is only kill-ratio dependent, the other is kill-ratio but also time dependent and thereby punishing in nature. The fitness evaluation is not very strict, as it does not punish factors such as: total movement, energy consummation or pattern specific behavior. The second fitness factor is meant to abstract predator efficiency and measure the killing of prey in the shortest amount of time. The factor only measures efficiency when prey is killed, not every time step, so if a prey kills prey only in the beginning, but not later, this would still yield some efficiency. If all prey were killed in zero time, this would yield a perfect score, but if all prey gets killed in the last time step, no score is applied to the second fitness, resulting in a total score of 50. The combination is used so that the wanted behavior is to kill all prey, preferably as early as possible.

$$killratio = \frac{killed}{preys} * 50$$
$$timeratio = 1 - (\frac{time}{totaltime})$$
$$efficiency = (killratio * timeratio) * 50$$

$$fitness = (killratio + efficiency)$$

3.4.2 Selections

The implementation uses a set of relatively trivial selection algorithms to simplify the evolution process and reduce the amount of computations. The selections always automatically select the best candidate solution, favoring strong elitism.After fitness values are assigned to candidate solutions representing the performance of generation of predator groups, the adult selection decides which that survives and grows up. The best solution is automatically transferred into the next generations candidate parents, so is the rest of the solutions performed above average in their generation. If only the best survived, the second best is also added to the group of candidate parents. The next is the parent selection, the best solution is assigned as father and random other solution is assigned as mother, a solution cannot be both father and mother. This efficient couple will produce the entire set of children solutions in the children crossover algorithms.

1. **Fitness evaluations:** Perform fitness evaluation and Sort all candidate solutions after performance.
2. **Adult selection:** Keep the best and all above average. Add the second best if only the best is added.
3. **Parent selection:** Assign the best as father, and a random above average as mother.
4. **Reproduction:** Create enough children to fill up the available space
5. **Crossover:** Perform random crossovers with both parents on all children.
6. **Mutation:** Perform random mutations on all children.
7. **Replacement:** Replace the old generation with the new, consisting of the new children, the best and above average solutions.

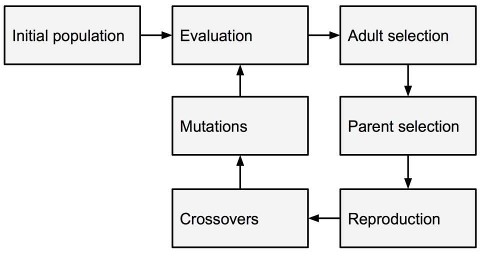

Figure 3.3: Simulated evolution.

3.4.3 Crossovers and mutations

The process of creating a new child solutions starts with evaluations and fitness selections, the selections result in mother and father solutions, who together create new children solutions. Children is in this implementations is created by cloning the father solution, then inheriting some genes from their mother using crossover, and at last provided with random mutations to differ from their parents. Crossover and mutations occurrences are based on two probabilistic rate variables and JavaScript's pseudo random Math.random function. Because this is a GA, the probability of evolutions through mutations is higher, than with crossover Spears [28]. Changing the mutation- and crossover-rates can change this if other evolution process is wanted. The children solutions genotypes are loop through, if a generated random number is within the crossover rate, the value on current index value is replaced by its mother current index value. Similarly, the mutations loops through the solutions, and a small value 0.1 is either added or subtracted from the current index value if a random number is within the mutation rate.

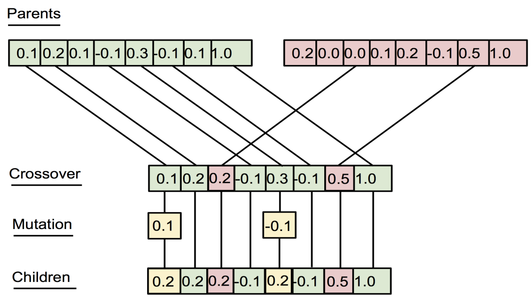

Figure 3.4: Genotype crossover and mutation

3.5 Artificial neural network

The artificial neural network (ANNs) used by each individual predator in the groups is based on Feedforward networks, as described in [35]. The ANNs don't use memory or recurrence in the networks. This simplification is done because of the increased system complexity caused by the number of agents and inputs. However, the implementation has more information and should be able to take more intelligent decisions than simple rules would produce. In ANNs, every agent's relative observations of nearby predators, prey and walls are input information. Agents also have the possibility to chose between 12 or none actions, deciding if and where they should move relative to their surroundings. Groups of predators can use this information to employ strategies and tactics when hunting prey, displaying intelligent behavior. For this ANN to work, the ANNs to harmonies inputs and output, through a network of weighted neurons, evolved by using GA. The implementation uses a total of 72 neurons, needing 72 weights. 36 sensor input neurons for different object types; the sensor neurons are combined into 12 input neurons in an input layer, one for each of the 12 vision zones. The input layer neurons are fired and crossed further to 12 neurons in the hidden layer. The last process is repeated once more two the 12 neurons in the output layer. Here, the strongest signal is sent to the predator as an action, deciding where to move. If no signal is strong, but a signal is very weak, another action is sent instead. If no action is wanted, the ANNs will return a -1 action: deciding that the predator should not move. Each firing signal is sent through an activation function, normalizing the signal before signals aggregate at the next neuron. This is an abstraction of chemical processes inside neurons in in the central nervous system, ensuring that only some impulses propagated further and result in actual actions. Each of the neurons are weighted by numerical values between -1.0 and 1.0. The weights are directly mapped from genotypes. Good or interesting weights are saved in weights.js.72 neurons can be used at every time step, for every predator. By an entire lifetime of 500 timesteps for a small group of 10 individuals, there could be 360.000 neurons firing. And if these was only one of 1000 generations of 10 candidate solutions. The numbers of firing neurons would be:$72 * 10 * 500 * 1000 * 1000 = 360.000.000.000$

3.5.1 Activation function

The implementation use a sigmoid function to decide how strong signals should be activated and fired further into the ANNs. 0.5 is subtracted from the result, so only strong signals above 0 propegate and combined at the next neurons. $signal_0$ is sent from its neuron through the following sigmoid function:

$$signal_1 = \left(\frac{1}{1+E^{-signal_0}}\right) - 0.5$$

$signal_1$ is fired further to the next 3 neurons, possibly combining with two other signals who have been through the same function.

The illustration show the ANNs model used in this project. It feeds 36 inputs for predator, prey and walls(red, green and blue) to create a input layer(yellow), the input neuron is mapped to a hidden layer(dark green), which is again mapped to an output layer(dark red). The output layer creates an output action(purple node). Each of the 72 neurons are weighted with the respective weights, at each layer node, the previous inputs are passed through an activation function, deciding if they should propegate further. The circular cross mapping allow vision impressions in nearby vision zones to greater affect the outcome in others.

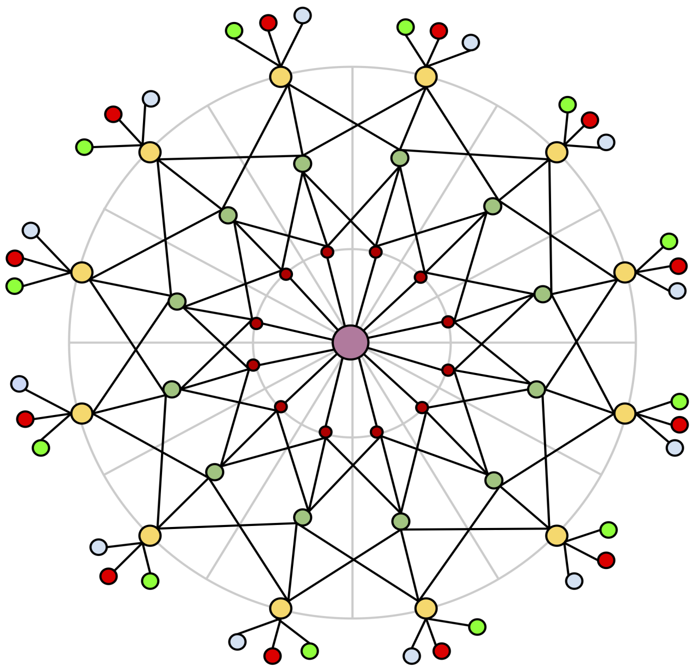

Figure 3.5: Artificial neural network model with 72 neurons.

Chapter 4

Experiments and Results

This chapter will experiment with the implementation to investigate research goal and questions. The investigation will use evolutionary algorithms and ANNs to create SI behavior in predator-prey scenarios. This approach should be able to produce more interesting, effective and creative behavior than using simple rules as approach. The chapter will lastly discuss implementation limitations and what could have been done differently.

4.1 Experiments

Results will be found by conducting experiments and observations related to research goal and questions. Experiments consist of multiple grouped simulations, each related to a specific purpose or separate research questions, but with the general research goal in mind. Discussion of results will be done directly after the experiment is completed. Common limitations will be discussed after the experiments.

To provoke discussion and gain better understanding of the results, hypothesis always will be made before any are conducted. I first have to establish that the GA-ANN implementation works and can successfully evolve SI. It will be proved by first evolving an over-fitted behavior from a simple genotype in the most basic scenario possible. The same simple genotype will be used to evolve more normal SI behavior by increasing the number of scenarios and complexity. And lastly evolve a manipulated SI behavior using the same setup as the normal SI behavior and manual genotype manipulation. The three experiments will be compared with simple rules in identical scenarios. The results will be used as a proof that the implementation works. The follow three experiments: Scaling group, invisible walls and environmental noise is setup to investigate aspects SI behavior, with the respect to research questions. Behavioral pattern observations will be done, by studying two visualizations of predator-prey scenarios, where predator employ evolved GA-ANN. The visualizations will be recorded as evidence. This combination of experiments and observations will provide me with deeper insight towards my goal: Gain understanding about how evolved predator swarm intelligence can be used instead of simple rules.

All experiments will be conducted in controllable simulated environments where most aspects can be measured. The Overfitting experiment will only be done one predefined scenario. The normal and manipulated experiments will be done with predefined, generative and random scenarios. Scaling group, invisible walls and environmental noise experiments will be done in a simulation as the overfitting experiment, except for using a different generative scenario and small variation in attributes: group size, wall vision or environmental noise.

Result discussion given in each separate experiment, it will be derived from the measure results simulations yields; the number of simulations in each experiment would have to be significant to ensure the results are not just lucky or cherry-picked. The number of simulations necessary to investigate each experiment will be set to 20 simulations, from these simulations, an average performance can be observed and discussed. Standard deviation will also be calculated and visible as thin grey error lines on the average fitness. The complete results are in located in large excel files and will be included with the code.

The experiment overview is as follows:

1. Overfitted experiment
2. Normal experiment
3. Manipulated experiment
4. Scaling group experiment
5. Invisible walls experiment
6. Environmental noise experiment
7. Observation study

4.1.1 Overfitting

Plan

Begin to establish that the GA-ANN works by evolving over-fitted behavior from simple initial genotypes in one scenario, as simple as possible. The results will be compared with using simple rules.

Setup

All predators start at the coordinates 300x250y A and prey at 100x250y B, called Scenario A in the appendix. Predators will need to learn to move left toward prey. The initial genotype weights used as seed is called initial (W1 in appendix), it sets predator, prey and wall weights to 0.1 and input hidden and output layer to 1. It will use the follow simulation settings:

Simulations:	20
Runtime:	10 minutes
Time steps	500
Generations	100
Population	5
Genotype	W1 (initial)
Mutation rate	0.5
Crossover rate	0.1
Predators position	A
Scenarios	B
Wiggle	False
Noise	False

Table 4.1: Experiment overfitting setup

Predator ANNs have identical initial weights for predator, prey and walls in this experiment. Which means their attraction to predators and prey are identical, and that evolution is necessary to learn hunting. Predators need to develop a strategy that enables them to prefer and hunt prey. The necessary learning process is reinforced by evolutionary pressure through fitness evaluations and natural selections.

Baseline

The simulations will use simple rules as a baseline for comparison. Predators using simple rules will always receive a fitness score of 90.7 in an identical scenario, no random variables is used, so this will always happen. The baseline results can easily be reproduced using the same parameters as above and the simple rules described in implementation chapted. Selected by setting useANN to false in simulation.js.

Hypothesis

Predators will successfully evolve a strategy to kill prey; they will be able to reach a fitness score of 90, proving them almost as successful as simple rules (90.7).

Result

Experiments show that average solutions reach a fitness score of 90.785, matching the performance of the simple rules. The highest recorded fitness score was 92.3(!), beating the simple rules by 1.6. Standard deviations were only 2.19 at generation 100. This is very low, meaning all generations were performing very well.

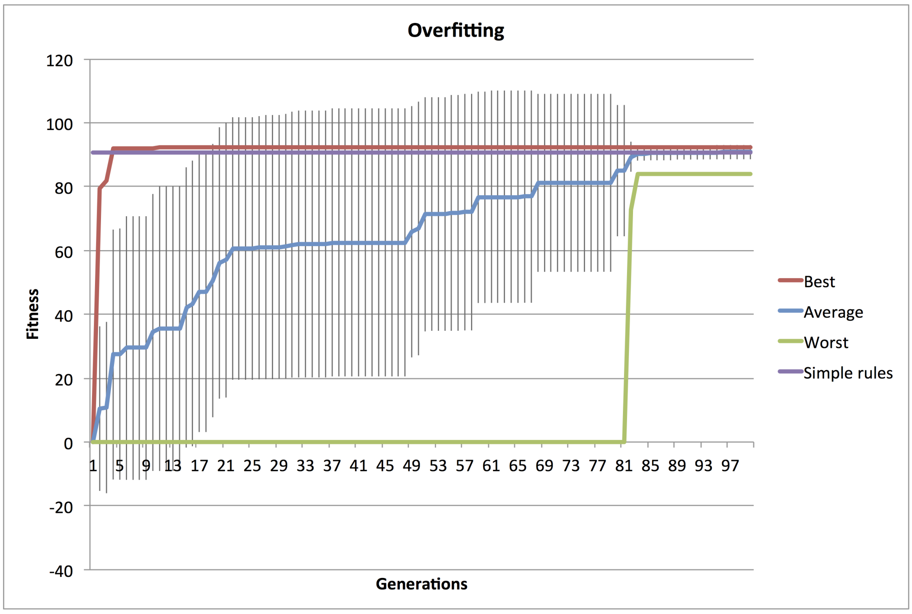

Figure 4.1: Experiment overfitting fitness landscape

Study of the fitness landscape of provided some interesting aspects of the simulations, described in chronological order:

- **Generation 1-22:** The average solutions where in a growth phase, steadily climbing and improving performance until they reach a fitness of 60 at the age of 22 generations, then they faced a temporary stagnation.
- **Generation 23-49:** The stagnation process continued until generation 49, then they finally began improving fitness again. One possible explanation to is that because the fitness function both reward on total kills ratio and killing/time ratio efficiency, the solutions first learned to kill, then later evolved swarming behavior like stigmergy or herding to approach and surround prey from different angles quickly.
- **Generation 49-85:** The average fitness is advancing with big improvements at the time, creating a platform structures in the fitness landscape. A possible explanation could be that the different simulations predators are learning how to kill faster by trying similar approaches at a similar time. Because the platforms are clearly visible and distinct in the fitness landscape, its probable that they simulations improve similarly at the same time. Improvements at different generations would smooth out the fitness landscape.
- **Generation 85-100:** The fitness landscape is flatting out after 90 and standard deviations are decreasing to 2.19. It still growing slightly, but it has almost peaked and cannot grow much longer.

The implementation proved the GA-ANNs approach could successfully evolve over-fitted predator swarm intelligence in this simple scenario. The average fitness values started at zero and improved gradually to over 90, which is a successful evolution of behavior. The best scores quickly became over-fitted and the worst struggled for a long time, but eventually reach the others. This is typical in overfitting scenarios, as it is essentially an easy problem to solve and should be possible for all to solve.

The hypothesis however, was disproven by the results. This is impressive, but surprising because the scenarios is very simple, and it is reasonable to believe the simple rules will perform better in this scenario, because the task can be accomplished by moving in a straight line towards prey. A possible explanation is that the simple rules implementation gives predators a slightly slower move speed as it's calculated differently. The speeds appear to be identical to visual observations, but it should be noted to future work that speed comparisons would have to be done more thorough. This problem only is critical in simple scenarios, where small speed differences can make or break the hypothesis. Much less so in complex scenarios, where effective swarming behavior can yield big differences and improve efficiency dramatically.

4.1.2 Normal

Plan

Explore the capabilities of the GA-ANN by the same initial weights as the over-fitted experiment, but with added random wiggles and multiple different scenarios. This is done with the goal to create a predator SI behavior that can prove efficient in different scenarios. This experiment challenges the implementation to greater extent than the over-fitted experiments, because it will use more of its features to evolve a more intelligent behavior. The results will be compared with using simple rules as a baseline.

Setup

The initial genotype weights are identical to those used in the overfitting experiment: 0.1 is used for prey, predator and walls, input, hidden and output layers weights are set to normal (1), to allow full propagation at default. The weights are in the appendix as custom W1.
The simulations will each use 8 different scenarios: Predators start at the position C in the center of the environment with coordinates 250x250y every run. Prey start at 8 different initial scenarios: 1 predefined, 4 generative and 4 randomly scattered.

Simulations:	20
Runtime:	1 hour and 40 minutes
Time steps	500
Generations	100
Population	5
Genotype	W1 (initial)
Mutation rate	0.5
Crossover rate	0.1
Predators position	C
Scenarios	B, D, V100, V400, H100, H400, R1, R2, R3, R4
Wiggle	True
Noise	False

Table 4.2: Experiment normal setup

Baseline

The experiment will use simple rules as a baseline for comparison. Simple rules also use the wiggle feature and will therefor perform differently every run, resulting in different fitness scores. An identical experiment is therefor conducted with useANN set to false, and simple rules enables. The simple rules experiment average is visible in the fitness landscape for comparison. Extended results are in the included files

Baseline Results The baseline fitness scores were stable around 40, on average 40.34. This was surprisingly high, considered the multiple different scenarios. The stable results is due to no random variables were used, except for the random wiggles every timestep.

Hypothesis

Normal simulations will perform better than the simple rules; predators will evolve swarming behavior that allows them to benefit from their power in numbers.

Results

Hypothesis disproved, the normal simulations scored on average much lower than the simple rules scores. They scored on average 5.15, while simple rules score 40.34. The highest recorded average score was 7,14, and lowest 0. The fitness landscape clearly shows an evolution process, slowly facing stagnation. The landscape provide some interesting aspects by inspection:

- The highest fitness values are increasing from generation 89 to generation 100, suggesting that the behavior could possible evolved further, if given the opportunity to continue.
- Standard deviation is very similar during the entire experiment, and never faced stagnation as in the overfitting experiment.
- The lowest fitness scores were more fluctuating after generation 69 and towards the end, meaning all candidate solutions were performing better than the initial genotype.

The combination of these aspects imply that the simulations were still evolving at generation 100, and that it could possibly have perform better if given enough time. 100 generations was probably to few generations for the experiments to reach its optimal performance. This was surprising and valuable information, noteworthy for future work. The wide combination of different scenarios made evolution of SI behavior a much bigger challenge without a doubt. But the results don't show if the challenge was made any harder, of it just needed more time. This forms an important question for future works.

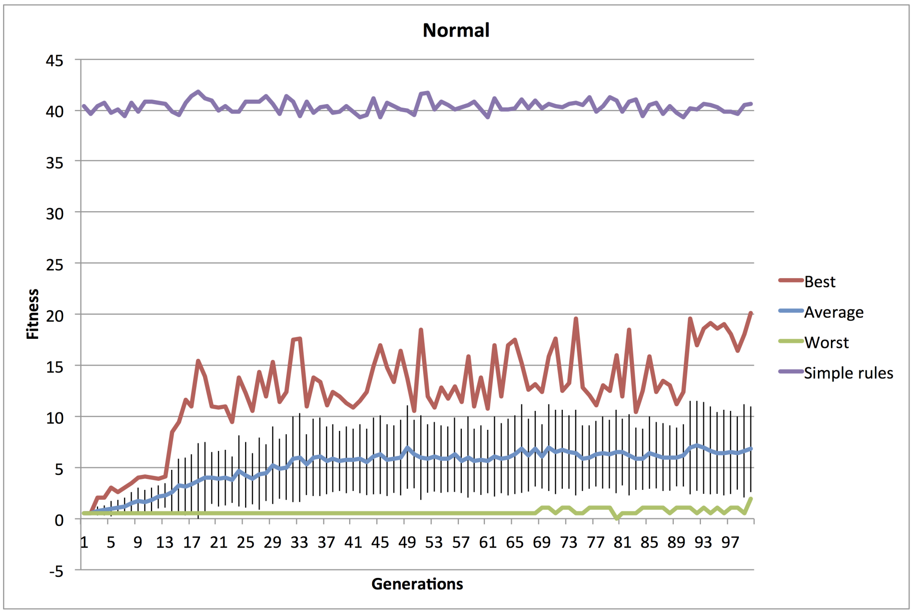

Figure 4.2: Experiment normal fitness landscape

4.1.3 Manipulated

Plan

Extend the Normal experiment further by using manual genotype manipulation. The results will be compared with using simple rules.

Setup

The setup is identical to the setup used in the normal experiment, with the exception of using manual genotype manipulation before starting simulations. The setup is manipulated to create the best possible starting point for evolution of efficient SI behavior. Genotype weights are manipulated at custom to be very attracted to prey (0.5), and to avoid nearby predators (-0.1), walls are visible, but no weights are set to prefer attraction or distraction (0.0). Input, hidden and output layers weights are set to normal (1), to allow propagation at default. The weights are in the appendix as custom W2.

Simulations:	20
Runtime:	1 hour and 40 minutes
Time steps	500
Generations	100
Population	5
Genotype	W2 (manipulated)
Mutation rate	0.5
Crossover rate	0.1
Predators position	C
Scenarios	B, D, V100, V400, H100, H400, R1, R2, R3, R4
Wiggle	True
Noise	False

Table 4.3: Experiment mainpulated setup

Hypothesis

The manipulated approach will perform better than the simple rules; predators will evolve swarming behavior that allows them to benefit from their power in numbers.

Results

Hypothesis proven. The average performance was stable around a fitness score of 61. It started at 61 and ended at 61, with maximum at 62 and minimum at 59.46.

This was surprising high and stable results. Even though the hypothesis got proven as predators performed better than simple rules, it has an important flaw. Predators did not learn to benefit from their power in numbers through evolution, the behavior were already there, and did not improved or adapted any further. The manipulated genotypes provided predators with a robust self-organizing, flocking behavior. Genotypes were altered to slightly prefer prey and avoid nearby predators if they came very close and no prey were close. This SI behavior proved successful in a large (8) group of scenarios. It is very likely that it is not the optimal solution, and that even better solutions can be found. This makes it very attractive to evolve further on the genotypes, or at least draw inspiration from when manipulating others. The experiment proves how good the implementation can perform if given enough time, freedom and evolutionary pressure to evolve. It is evidently better than simple rules in the group of scenarios.

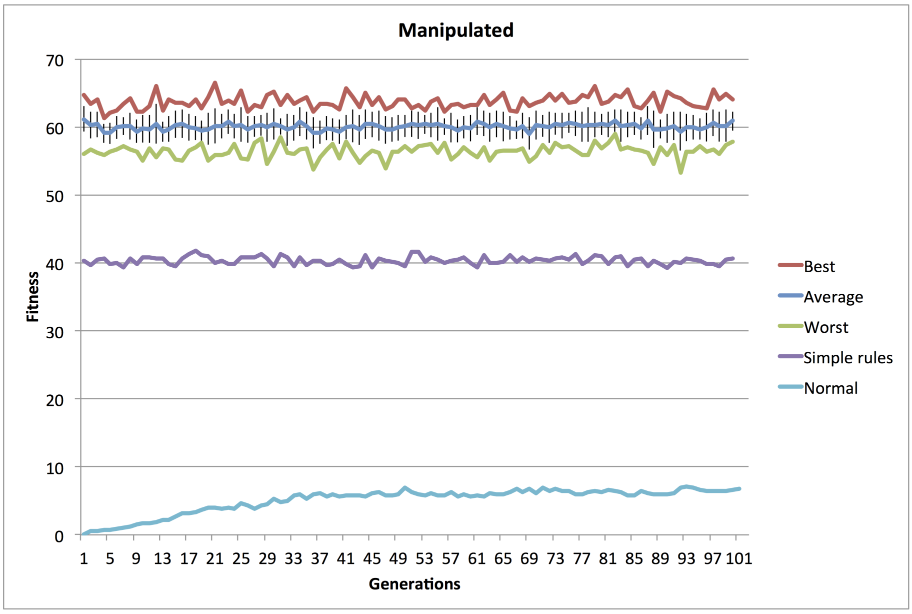

Figure 4.3: Experiment manipulated fitness landscape

4.1.4 Scaling group

Plan

Experiment with different group sizes to investigate if SI behavior evolved for a defined group size can benefit from an increased group size. The experiment will evolve a normal SI behavior as baseline, then alter the group size and investigate if the results change.

Setup

The baseline evolution setup is identical to the setup in the overfitting experiment, except for using a different scenario, where prey is generated along a vertical line. This evolution is used to evolve a normal SI behavior to be used as a baseline. The experiment will scale the amount of agents by 2x, 4x, 6x, 8x and 10x, both predator and prey. This will help to investigate if they maintain the same fitness by running single simulations and investigate each time step. The experiment does not involve any random elements, and will therefor be identical each time; it is not necessary to run 20 simulations because the result cannot change.

Simulations:	20
Runtime:	10 minutes
Time steps	500
Generations	100
Population	5
Genotype	W1 (initial)
Mutation rate	0.5
Crossover rate	0.1
Predators position	A
Predators	5, 10, 20, 30, 40, 50
Preys	20, 40, 80, 120, 160, 200
Scenarios	V100
Wiggle	False
Noise	False

Table 4.4: Experiment scaling group setup

Hypothesis

Fitness will increase as the numbers are scaled, because this is a closed environment and they can benefit from their numbers, even though they have not evolved to do so.

Results

Hypothesis disproven. Every fitness landscapes remained almost identical to the baseline; the scaled groups did not increase or decrease the fitness scores. This lead me to believe predators have to learn to use their power in numbers. It could be interesting to evolve a group of predators with constant variations group size. The variations could let predators evolve different intelligences to be used when the same size appear again.

4.1.5 Visible walls

Plan

Experiment with different wall visibility to investigate if SI behavior pre-evolved for a defined scenario can be used in environments where the walls are less or not visible. The experiment does not include any random events, and will not benefit from multiple simulations.

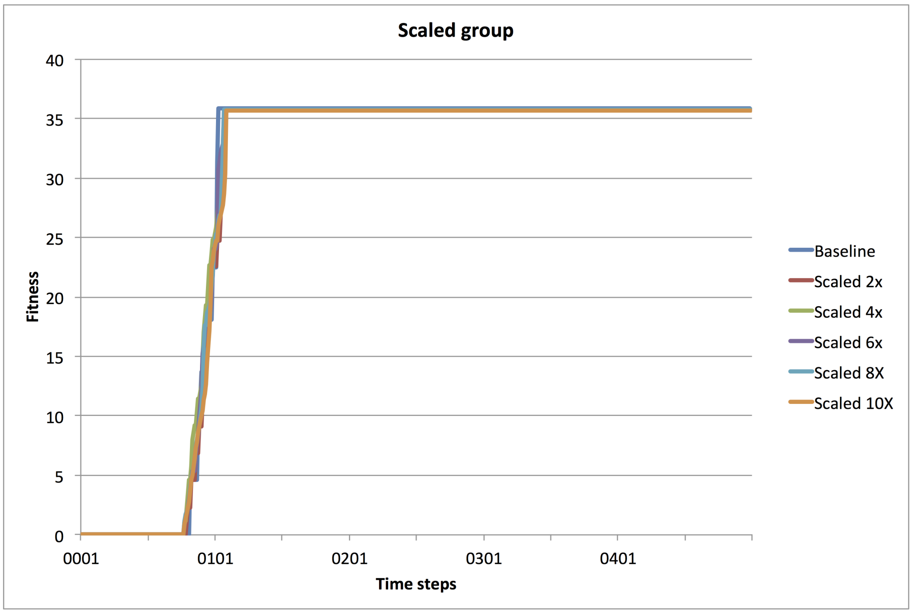

Figure 4.4: Experiment scaling group performance

Setup

The setup is identical to the setup in the scaled group experiment, but with constant agent numbers and a lower wall visibility.

Baseline

Baseline is the identical to the one used in the scaling group experiment.

Hypothesis

Predators will perform considerably worse, as they have a GA-ANN that's adapted to benefit from the environmental walls as obstacles used to constrain prey.

Results

Hypothesis proven, performance became considerably worse. The performance began decreasing slowly until it reached 10%, the fitness scores dropped to zero after that. The beginning can be explained with the impressions the walls contributed with, increased the input from prey and stimulated the hunting phase.

Simulations:	20
Runtime:	10 minutes
Time steps	500
Generations	100
Population	5
Genotype	W1 (initial)
Mutation rate	0.5
Crossover rate	0.1
Predators position	A
Predators	5
Preys	20
Scenarios	V100
Wiggle	False
Noise	False
Wall visibility	75%, 50%, 25%, 20%, 15%, 10%, 5%, 0%

Table 4.5: Experiment visible walls setup

With the reduction of walls impressions, the stimulations were also reduced. The catastrophically decrease after 10% can be due to that the ANNs have evolved to be used in the closed environment. When the surrounding environment suddenly disappears, the ANNs seize to work as well, as prey impressions alone is not strong enough to fire neuron responses. If the swarm had trained on similar scenarios, they might have learned and evolved behavior that could flock around searching, only by using local interactions with each other. The disappointing lack of this behavior could be due to the simple evolved genotypes, only created to survive in one overfitting scenario.

4.1.6 Environmental noise

Plan

Experiment with environmental noise added to the ANN inputs as random events.

Setup

The setup is identical to the setup in the scaled group experiment, but with addition random noise. Extra noise will be added to predator, prey and wall inputs. Noise will be created by using JavaScript random function, creating a number between 0 and 1, and then divided by 50 to make it smaller. Regular single agent inputs are between 0 and 1.

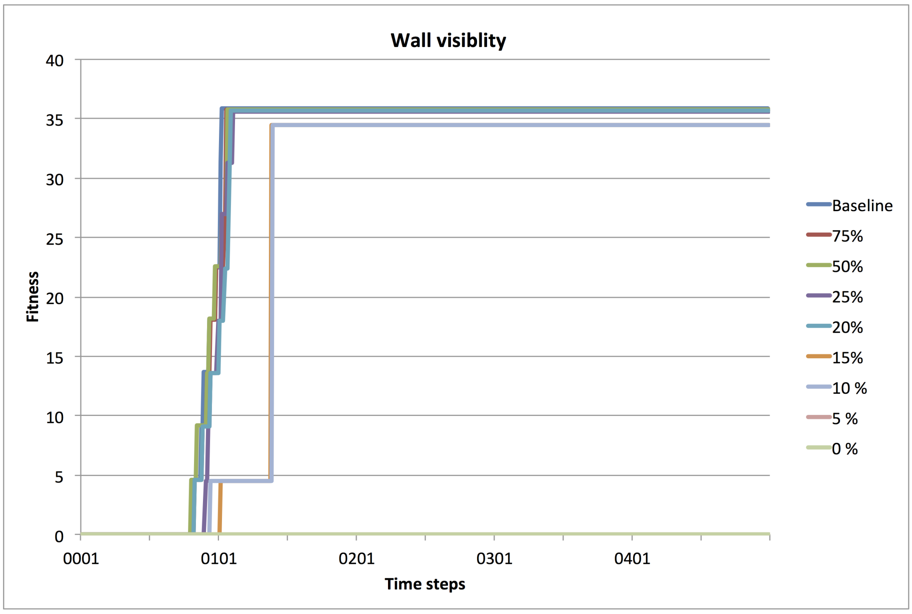

Figure 4.5: Experiment visible walls performance

This simulation includes random variables; it will therefor be conducted 20 simulations.

Baseline

Baseline is the identical to the one used in the scaling group experiment.

Hypothesis

Random environmental noise will affect the performance with a reduction of 50% in fitness scores.

Results

Proven, the fitness scores got reduced by ca 50%. The scores and standard deviation remained stable after timestep 110, same as the baseline. This could mean that both baseline and experiments have faced stagnation in a corner or situation - where they don't have evolved intelligent behavior for yet.

Simulations:	20
Runtime:	10 minutes
Time steps	500
Generations	100
Population	5
Genotype	W1 (initial)
Mutation rate	0.5
Crossover rate	0.1
Predators position	A
Predators	5
Preys	20
Scenarios	V100
Wiggle	False
Noise	Random number in the range (0.02 - 0.002)+/-

Table 4.6: Experiment environmental noise setup

4.1.7 Observation

This observation study will investigate two video clips of evolved GA-ANNs SI behavior and explain the observable patterns that emerge. Videos are supplied as an addition to the appendix.

Video 1

The scene is set in a typical, where 5 predators hunt 20 preys. It lasts 8 seconds.

- Predators eagerly flock toward a group of prey that splits into to separate groups; predator maintains the stable direction towards the center.
- All predators decide two pursue the hunt for the group with the highest density of prey.
- Predators catch and kill most prey in the big group; only one prey manages to escape. Predators split up, three goes after the closest one: The escapee. The other two starts to search for the prey that flew earlier.
- One predator manage to herd prey in the upleft corner, managing to kill the prey alone, before he quickly joins his predator peer in the hunt for the one escaping at the top. Meanwhile, three predators are flocking together, chasing the escapee, before one decides to stop, as the predator group becomes dense and it's not necessary for him to contribute. His friends herd the prey into the downright corner and successfully kill him off.

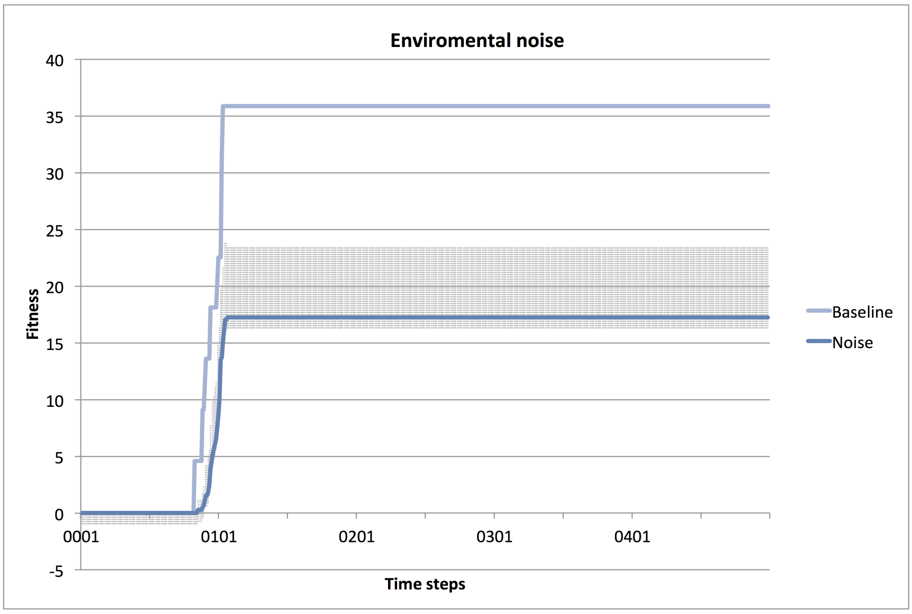

Figure 4.6: Experiment environmental noise performance

- The last living prey are fleeing along the right edge as he tries to escape from the predator, persistently lurking behind him, while making sure he always stay one step behind so the prey don't have anywhere to escape.
- Two predators collaborate to encircle, capture and kill the last prey.

Video 2

The scene is set in a Random scenario, where 5 predators hunt 20 preys that is scattered across the environment. It lasts 11 seconds.

- Predators start together but quickly hunt toward nearby prey, until all predators have formed a diagonal predator line structure. They use this structure to herd prey towards the left wall.
- One predator breaks the formation to hunt a kill a three prey that have escaped and hide in a corner. After this, he resume his position in the formation, pushing prey closer to the left wall.
- As one of the predators in formation decide its time to strike, creating a small gap, two preys see this and escapes through the gap. All predators

quickly stop what they were doing and chase after the escapees like a flock, then quickly forming a new line structure before they continue.

- The line structure moves towards the escapees and herd other prey toward the upright corner, making them easy to kill.
- Meanwhile, a lone wolf predator is killing two preys near the left alone.
- The group engages on the corner, killing all but on. One prey manages to escape but on single predator sees it and begins hunting after him. He succeeds.
- The group and lone wolf divides up into small groups that hunt down the last remaining prey.

Results

The videos clearly show signs of eager active behaviors in some predator phases, and some more cold and calculating behavior in other phases. Predators are flocking together to hunt for prey, but self-organize when multiple groups can be captured. Predator herded prey into corners, in groups and alone. They see each other and sometimes decide not to engage if nearby predator have better chances; this is due to stigmergy as global structure is formed through local interactions. This is also visible when predators form the line structure, created to herd multiple preys into a wall or corner. When prey escape outside the structure, predators quickly respond by chasing escapees. Predators don't rush when they know they are going to capture prey at last, this is similar to persistence hunting. The behaviors are robust, as one or few individuals can successfully capture prey, although more is better.

4.2 Limitations

Although I am satisfied with the implementation and the results it yielded, several flaws or limitations become apparent after running experiments.

4.2.1 Speed

The overfitting experiment showed that GA-ANNs agents were possibly slightly faster than the simple rules, not by much, but still faster. This is a flaw that was not supposed to happen, because the approaches were though to be equal in this sense. Luckily, the flaw will only have significant impact on very simple experiments like this, which could be compared with something like: learn how to drag race in equal cars and beat the opponent with one second. The flaw is less important in bigger experiments; where strategy and tactics play a bigger role in determine performance. The flaw however should motivate future work to design performance comparisons of different solutions; this would be better, more stable and number-driven way.

4.2.2 Environment

The environments plays a big role in experiments and the performance of new solutions, this should be made easier to alter and change, so that solutions can learn from a bigger pool of different environment. It quickly become too difficult to alter the environments in a big way, this was an obvious limitation. Physics, shapes, terrain, friction, obstacles, noise are all interesting aspects that would benefit the complexity of evolved intelligence, if given enough time. The implemented environment was initially design to work in three dimensions, but the task quickly grew to big for me to incorporate a full 3D world with all aspects and still focus on the predator-prey relationship. Future work should try to devote more attention to this, and experiment with scoping down the solution and scoping up the environment. The solution and environment will always be paired together, and a more complex environment will need a more complex solution eventually. Environment should not be a limitation, but rather an reason, enabler and facilitator for evolution.

4.2.3 Simple rules

The experiments only compares to one set of simple rules. The rules only have two main objectives: 1: Escape the closest enemy if someone is visible, 2: move towards the center if nobody is visible. The rules are simple and effective, but more effective simple rules could most likely be written. A interesting exercise would be to try creating simple rules to always beat the evolved approach, this will be suggested for future work.

4.2.4 Vision

3D worlds would need a 3D spherical vision; this can be engineered by extending the existing 2D vision with an additional 12*12 = 144-12 = 132 vision zones or 132 * 3 = 396 vision inputs. The increased number of inputs would of course also scale up the ANNs equivalently. This scaled up vision model would not have been feasible with the used implementation, therefor simplifications had to be made to overcome the limitation. Another limitation concerning the vision model is the lack of dead zones/blind spot, as described in related work. It would have been interesting to investigate further how agent direction and a dead zone/blind spot would have affected performance. Selection algorithms were simplified to ensure that fit solutions survived, the approach favored strong elitisms, and this can possibly have led to limited SI creativity. Experiments with different selection algorithms would have been very time-consuming, but could have benefitted performance greatly. The system complex needed when designing swarms and following heavy experiments set a limitation on exploration of different evolution approaches. Multiple researchers working together on the same project would probably have done this more effectively, and could yielded better results.

4.2.5 Evolution

The GA produced less impressive results than imagined. The strong selection elitism and low number of candidate solutions may result in less solution diversity than wanted, and suppressed the right to live for new creative new solutions if they performance bad overall. This limitation could be experimented with further, by creating more selections algorithms and test multiple setup combinations to evolved more creative behaviors. More interesting genotypes can be used, and mapped to richer phenotypes. This could probably also make the challenge of keeping solutions that 'only' performs creative and good, not perfect. The combinations of these approaches can possibly overcome the limitations the GA has today, and produce more interesting behaviors.

4.2.6 Computer performance

Simplifications and abstractions done in the GA-ANNs implementation are necessary to evolved SI within the scope of this project. This is necessary because of the massive amount of calculations needed to evolve and use weights on a swarm of predators, not just one individual. Each experiment consisted of at least 20 simulations, each simulation contains numerous generations, each generations contains many solutions, each solution is tested in several different world scenarios, each solution contains a group of predators, each predator lives a short, but intensive life hunting preys using visual inputs to chose actions, each predator chose a action every time step. The amount of calculations and processing power needed to evolve SI behavior became an obvious limitation, which limited how many scenarios could be learned or how many generations that was used. The experiments were big and complex, but still feasible to calculate on a normal desktop computer. The implementation complexity is designed to run on a normal computer (2.4Ghz Intel 2 Cores, 8gb RAM) and used ca 1 hour and 40 minutes to complete a large experiment with 20 simulations. Bigger experiments would be to computationally expensive and not feasible with the current implementation.

Chapter 5

Conclusion

This chapter will provide conclusion on evolving predator swarm intelligence, review research questions, explain research contributions and suggest future work.

5.1 Conclusion

Effective predator-prey swarm intelligence can be engineered by using genetic algorithms and artificial neural networks. The combination is well suited for swarming problems. Problems where size, combinations and constant change makes it very complex to design simple rules. The employed vision model enables predators to capture enough environmental impressions in a satisfactory way; it can be built upon when designing vision models in the future. However, agent, ANN and vision model complexity have to be adapted to the environments, not the opposite. Simulated environment richness should receive more and earlier attention to avoid the evolution of handicaps in real complex environments. I humbly advise that the entire environments and assigned task should be modeled before agents and possible solutions are considered. In hindsight, this seems very obvious, but it is difficult to model an environment without accidentally consider possible solutions. A way of doing this is to assign different tasks isolated groups of researchers, so that solution designers do not know what the environmental designers are thinking, and visa versa. The similar approach could be used with teams designing separated simple rules and ANNs, each with the goal to prove their solution is the most efficient. I can also conclude that manual manipulation of agent properties such as genotypes can be an effective way of exploring and mapping of performance.

The research have not produced any earth shattering results, but still provided some focus and attention to areas in need: Predator-prey relationships, Swarm intelligence, GA-ANNs, simulated environments and vision models. The experiments proved that the implementation can perform better than the set simple rules used for comparisons, but this could only mean they were too simple and deserve to be developed further. Regardless of the fallout from that comparison, the research contributed by pushing the need for future exploring and evolution of swarm behavior, both simple and evolved.

5.1.1 Research questions

The following research questions was created and posed in the introduction chapter. Even though I only have scratched surface of evolving predator swarm intelligence, I now possess the knowledge to answer them with conclusions.

- **Research question I:** Can GA-ANNs be proved to sucessfully evolve SI behavior?
  Hypothesis: Yes, the implementation should be fitted to sucessfully evolve behavior.
  Conclusion: Hypothesis proved, the combination of the Overfitting, Normal and Manipulation experiment proves that it can be used to evolve SI behavior. Overfitting proved that it could be done, at least in easy scenarios. Normal shows that it's capable of evolving towards more complex behavior, but that it will need enough time to so, if the scenarios are many and complex. Manipulation proved that the GA-ANNs implementation is capable of solving complex problems if it uses good genotypes.

- **Research question II:** Will GA-ANNs benefit from manual manipulation of genotypes?
  Hypothesis: Yes, there will exist a combination of genotype values that result in good behavior. When we know which genotypes and weights are mapped together, we can remove all probabilistic factors and try manipulation until we find a combination that is beneficiary.
  Conclusion: Hypothesis proved. The Manipulation experiment proved that the implementation benefit from manual manipulation. Manipulations were done to favor prey and disfavor predators, this led to self-organizing flocking behavior, where predators moved together towards prey, while keeping a small distance between each other. This enabled the swarm to explore a bigger area and improve predation.

- **Research question III:** Does group size scaling of affect performance of previously evolved agents?

Hypothesis: Yes, A reduced amount of agents will become much less efficient because the predators are evolved to use their power of numbers to their benefit. Hence, the benefit and following performance will decrease. An increased amount of agents, benefit and a performance should increase some, but not so much as it will decrease. This is because the new larger amount of predators haven't learned to benefit from their numbers yet, both the number of predators in the group, and the number of available prey to be hunted. The neural input of more prey can seem overwhelming to ANNs evolved to find and hunt after few individuals.
Conclusion: Hypothesis disproven, The Scaling experiment showed that predators have to evolve SI behavior that can benefit from different scales; the increased scales did not improve the results. To increase solution robustness, this should be experimented with.

- **Research question IV:** Will the visibility of walls or environmental noise affected performance?
  Hypothesis: Yes, the environmental changes will have big impact on performance, because the GA-ANN is evolved in simple environments without these changes.
  Conclusion: The wall and noise experiments proved this hypothesis correct, at least in the simple scenario used. Results may have been different if other scenarios and combinations were used, but this is the topic of future work.

- **Research question V:** Will the evolved agents clearly display SI behavior?
  Hypothesis: Flocking, herding, stigmergy and self-organization can be observed.
  Conclusion: Hypothesis proven, all the behaviors can be observed in evolved predators, even though they have not been programmed directly to do so, as simple rules would do. They also showed persistent hunting phases and eager phases.

5.2 Contributions

The research has provided me with deeper knowledge about evolved predator swarm intelligence in simulated environments, insight I want to communicate to future researchers. The most important insight in this thesis is:

- The importance of analyzing simulation environment complexity.
- How vision models and artificial neural networks can be used by autonomous agents.

- The difficulty of evolving predator swarms using evolutionary algorithms.
- Observational study of swarm behavior using visualization of evolved behavior.
- Designing and comparing simple rules with evolved behavior

Researchers who study similar and related subjects will benefit from reading this thesis, especially with these insights in mind.

5.3 Future Work

This research has investigated evolved predator swarm intelligence. It has been an exciting and challenging journey that has led me places and showed by related research; subjects and ideas I didn't knew existed. Those areas deserve more attention, attention I didn't have the time and resources to afford and opportunity to pursue alone at this time. This is the most interesting future work I can suggest as future work:

- Evolve individual behavior in additional to group behavior.
- Employ several fitness functions for different behavioral traits.
- Flip the scenario; try to design simple rules to beat the evolved GA-ANNs approach.
- Create a simple rules competition, where the goal is to beat each others rules.
- Explore simulated environments with altering factors, physics and real terrain data.
- Replicate Spinner dolphins' prey-aggregation swarm behavior.
- Experiment with evolving encoding in ANNs to find better neuron mapping patterns.
- Automate swarm behavior pattern recognition by using machine-learning classification instead of manual observation.
- Investigate performance of predator-prey co-evolution

Bibliography

[1] Banerjee, V. and Banerjee, A. V. (1992). A simple model of herd behavior. *Quart. J. Econom*, pages 797–818.

[2] Beckers, R., Holland, O. E., and Deneubourg, J. L. (1994). From local actions to global tasks: Stigmergy and collective robotics. pages 181–189. MIT Press.

[3] Beni, G. and Wang, J. (1989). Swarm intelligence in cellular robotic systems.

[4] Benoit-Bird, K. J. (2008). Cooperative prey herding by the pelagic dolphin, stenella longirostri.

[5] Brooks, R. (1985). A robust layerd control system for a mobile robot. Technical report.

[6] Brooks, R. A. and Flynn, A. M. (1989). Fast, cheap and out of control. In *Artificial Intelligence Laboratory, Massachusetts Institute of Technology (MIT*.

[7] Carrier, D. R., Kapoor, A. K., Kimura, T., Nickels, M. K., Satwanti, Scott, E. C., So, J. K., and Trinkaus, E. (1984). The energetic paradox of human running and hominid evolution. *Current Anthropology*, 25(4):pp. 483–495.

[8] Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. pages 1–8.

[9] Dash, G., Weaver, J. N., Arroyo, A. A., Wright, S., and Schwartz, E. M. (2012). Orchestration of intelligent ground vehicles into a homogeneous swarm via the google cloud.

[10] Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART B*, 26(1):29–41.

[11] Ducatelle, F., Caro, G. A. D., Gambardella, L. M., Ducatelle, F., Caro, G. A. D., and Gambardella, L. M. (1999). Cooperative self-organization in a heterogeneous swarm robotic system.

[12] Garis, H. D. (1990). Genetic programming - building artificial nervous systems with genetically programmed neural network modules.

[13] Joyent, I. (2015). Node.js - http://nodejs.org.

[14] karl Sims (1994). Evolving virtual creatures.

[15] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization.

[16] Koza, J. R. (1997). Genetic programming.

[17] Lohn, J. D., Hornby, G. S., and Linden, D. S. (2006). An evolved antenna for deployment of nasas space technology 5 mission.

[18] Mataric, M. J. (1994). Interaction and intelligent behavior. Technical report.

[19] Miller, P. (2010). *The Smart Swarm: How Understanding Flocks, Schools, and Colonies Can Make Us Better at Communicating, Decision Making, and Getting Things Done.*

[20] (NECSI), T. N. E. C. S. I. (2015). Predator-prey relationships.

[21] Oboshi, T., Kato, S., Mutoh, A., and Itoh, H. (2002). A simulation study on the form of fish schooling for escape from predator.

[22] Olson, R. S., Hintze, A., Dyer, F. C., Knoester, D. B., and Adami, C. (2013). Predator confusion is sufficient to evolve swarming behaviour. *Journal of The Royal Society Interface*, 10(85).

[23] PBS, P. B. S. (2014). Nature: The gathering swarms.

[24] Reynolds, C. W. (1987). Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics.*

[25] Schuster, M., Paliwal, K. K., and General, A. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing.*

[26] Selfridge, O. (1959). Pandemonium: A paradigm for learning. *Mechanisation of Though Processes.*

[27] Shapiro, S. C. (1982). Artificial intelligence.

[28] Spears, W. M. (1995). Adapting crossover in evolutionary algorithms. In *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 367–384. MIT Press.

[29] Spears, W. M., Spears, D. F., Hamann, J. C., and Heil, R. (2004). Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17:137–162.

[30] Steel, L. (1989). Cooperation between distributed agents through self-organisation. Technical report.

[31] Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the SevenLh International Conference on Machine Learning*, pages 216–224.

[32] Threejs (2015). Three.js - http://threejs.org.

[33] Tollrian, J. M. J. R. (2006). Prey swarming: which predators become confused and why?

[34] Tomassini, M. (1996). Evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6:443–462.

[35] Trentelman, H. L. (1993). Representation and learning in feedforward neural networks.

[36] ul-hassan Usmani, Z. and Tariq, A. (2009). Influencing customers through customers – simulation of herd behavior in supermarkets.

[37] Wander Meer, R., Breed, M., Winston, M., and Espelie, K. E. (1998). Pheromone communication in social insects: sources and secretions. *Pheromone Communication in Social Insects: Ants, Wasps, Bees, and Termites.*

[38] Wood, A. J. and Ackland, G. J. (2007). Evolving the selfish herd: emergence of distinct aggregating strategies in an individual-based model. *Proceedings of the Royal Society of London B: Biological Sciences*, 274(1618):1637–1642.

[39] Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE.*

Appendices

5.4 Implementation

The implementation can be run in two different ways: Visualization or simulation. Visualization uses a predefined set of weights and a selected scenario. Simulation is used to evolve weights, and test performance of many scenarios. After a simulation run complete, the weights are visible in the terminal window, these can be copied into the weights.js file and used in visualization to study behavior.

It is build in mostly JavaScript, using a simple text editor. With the help of Safari, Node.js and a few others:

- Three.js - Used for visualizations.
- OrbitControls - qiao, mrdoob, alteredq, Westley Lang and erich666: used for camera controls in Three.js
- jQuery.js: Enables interactive content in web browsers, essential for Three.js
- colors.js, Used for color outputs in terminal output.

5.4.1 Visualization

Visualization is run by opening index.html in Safari, it starts immediately. It is possible to switch between ANN and simple rules live during the visualizations by clicking the visible buttons. The variables below can be experimented with:

```
var maxTimestep = 500;
var numberOfPredators = 5;
var numberOfPreys = 20;
var useANN = true;
```

```
    var wiggle = true;
    var noise = false;

    //Positions are located in the testing/positions.js

    var predatorsStartPositions = Positions.Center(numberOfPredators);
    var preysStartPositions = Positions.Random(numberOfPreys);
    predators.setWeights(Weights.manipulated());
```

5.5 Simulation

For simulations to begin, it is necessary to uncomment the follow lines of code. The require statements are necessary for simulations to work.

predatorController.js:

```
//var Agents = require('./agentController.js');
//var NeuralNetwork = require('./ann.js');
//var Weights = require('../testing/weights.js');
```

preyController.js:

```
//var Agents = require('./agentController.js');
```

agentController.js:

```
//var THREE = require('../testing/threemock.js');
```

After the comments are removed, the files should work as aspected and simulations can be performed. It can be runned by starting simulations.js in a command line window, it can be done like this:

```
$node simulations.js
```

All aspects of the simulations can be configured, to test a spesfic hypothesis. But different variables are located in different files. These are the most important variables and settings to explore.

simulation.js:

```
var numberOfPredators = 10;
var numberOfPreys = 40;
var timesteps = 500;
var generations = 100;
var weights = Weights.manipulated();
var worlds = [];
var numberOfExtraWorlds = 0;
var useAnn = true;
var wiggle = true;
var ANNnoise = false;

function addPredefinedAndGenerativeWorlds() {
    worlds.push(Positions.Left(numberOfPreys));
    ...
}
```

ga.js:

```
this.mutationRate = 0.6;
this.crossoverRate = 0.1;
this.populationSize = 2;
```

predatorController.js:

```
this.visionRange = environment.width-200;
```

agentController.js:

```
this.killDistance = 2;
this.wiggleRoom = 0.2;
this.speed = 1;
```

5.6 Weights

These are the genotype weights used in experiments. The weights.js file contains more weights and functions to create random weights and sets of identical ones.

Initial

```
[
    //Prey
    0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
    //Predators
    0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
    //Walls
    0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
    //Inputs
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    //Hidden
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    //Outputs
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
]
```

Overfitted

```
[
    // Prey
    0.2, 0.1, 0.1, 0, 0, 0.1, 0.1, 0.4, 0.20000000000000004, 0.1, -0.2, 0.2,
    // Predators
    0, 0.2, -0.1, -0.1, 0, 0.2, 0, 0.30000000000000004, 0.1, 0.2, 0.1, 0.1,
    // Walls
    -0.2, -0.1, 0.30000000000000004, -0.2, 0.1, 0.2, 0, 0.2, 0.2, 0, 0.1, -0.3
    // Inputs
    0.9, 0.9, 0.9, 0.9, 1, 1, 0.8, 0.7000000000000001, 1, 1, 0.9, 1,
    // Hidden
    0.9, 1, 0.8, 0.8, 0.9, 1, 0.8, 0.9, 0.9, 0.9, 1, 1,
    // Output
    0.9, 1, 0.8, 0.9, 1, 1, 0.9, 1, 1, 0.9, 0.9, 0.6000000000000001
]
```

Manipulated

```
[
    //Prey
    0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
    //Predators
    -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1,
    //Walls
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    //Inputs
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    //Hidden
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    //Outputs
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
]
```

Scaling baseline

```
[
    0.2,0,0.1,0,0.1,0,0,0, -0.2,0.1,0.2,0,0.1,
    0.30000000000000004,0.1,0,0, -0.2, -0.1,0,0.30000000000000004,0,0,0.2,0, -0.1,
    0.2, -0.1,0.2,0.1,0.2,0,0,0.1,0.2,0.1,1,1,
    1,1,0.9,1,1,1,0.8,1,1,1,1,1,
    1,0.9,0.8,0.8,0.9,1,1,1,1,0.9,
    1,0.9,1,0.8,0.8,0.9,0.9,1,1,1,1,1

    ]
```

5.7 Scenarios

These are the scenarios used in experiments. The positions.js file contains more predfined positions and functions to generate a infinte amount of random scenarios.

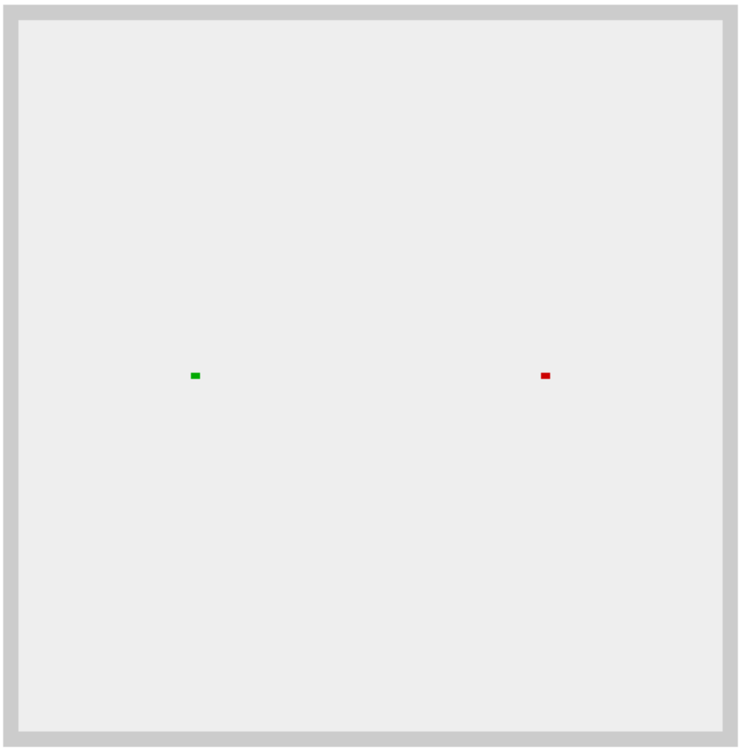

Figure 5.1: Scenario: Predator:A - Prey:B

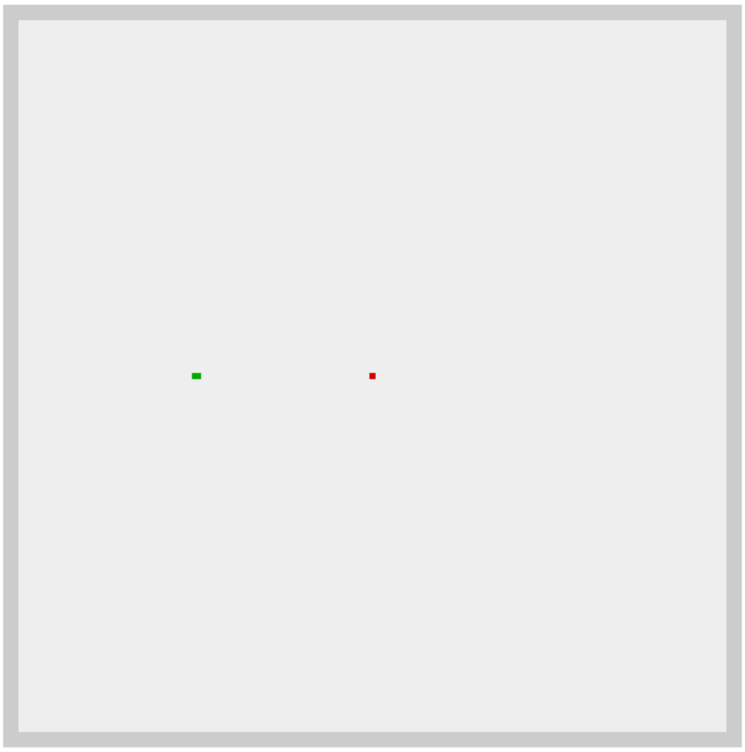

Figure 5.2: Scenario: Predator:C - Prey:B

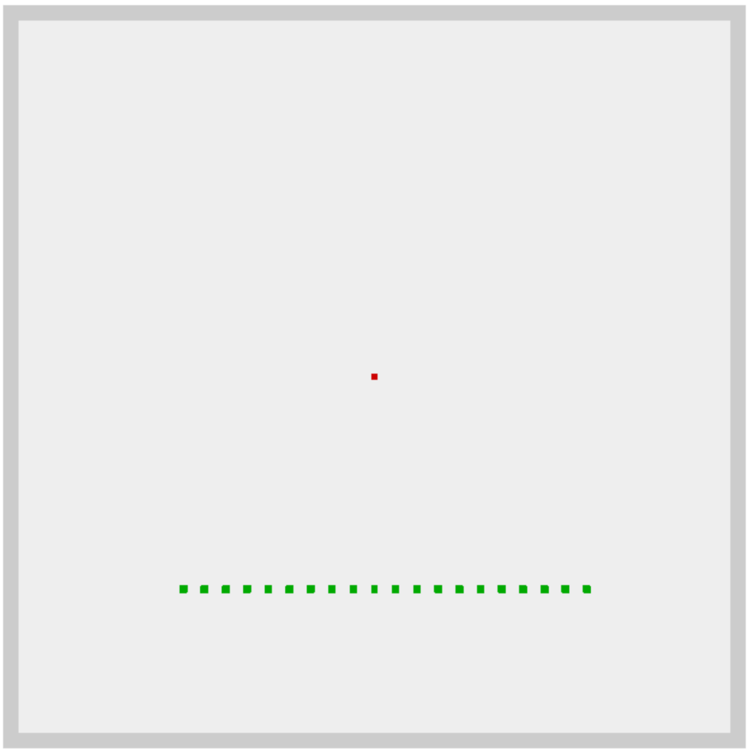

Figure 5.3: Scenario: Predator:C - Prey:H100

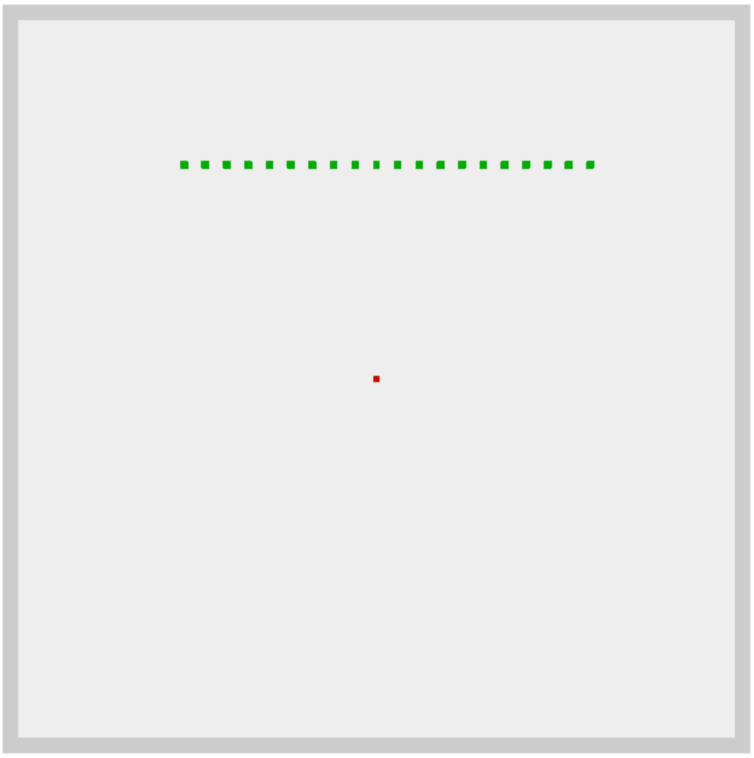

Figure 5.4: Scenario: Predator:C - Prey:H400

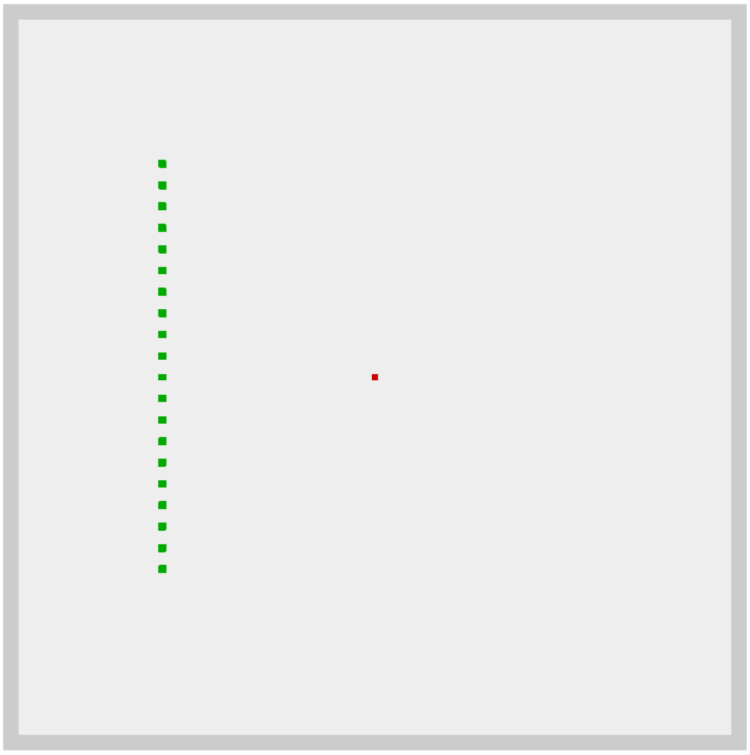

Figure 5.5: Scenario: Predator:C - Prey:V100

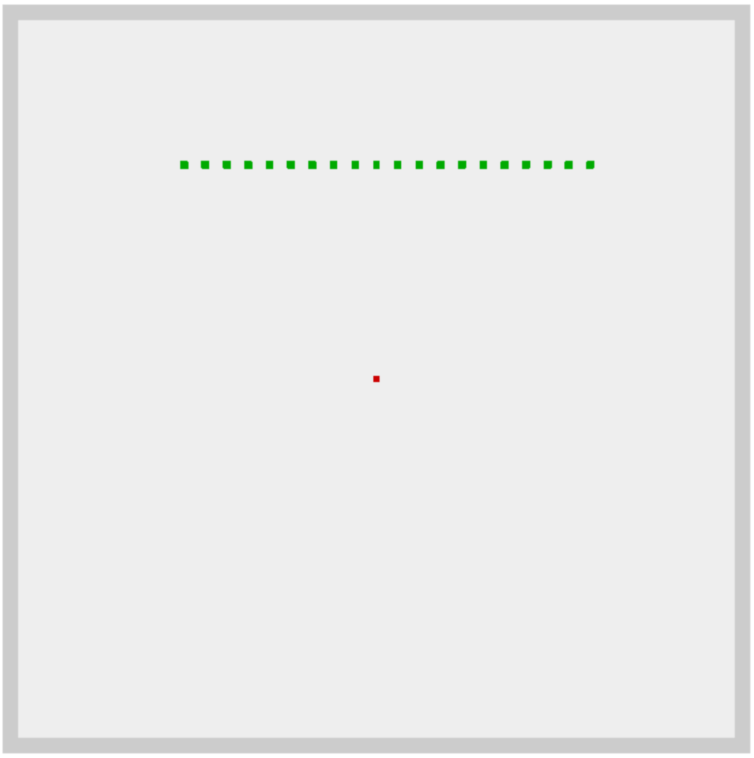

Figure 5.6: Scenario: Predator:C - Prey:V400

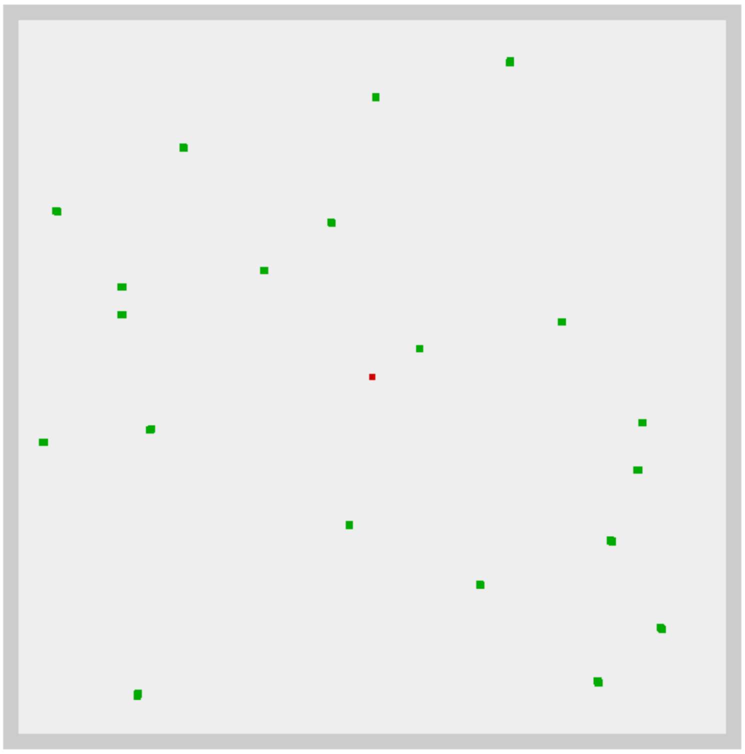

Figure 5.7: Scenario: Predator:C - Prey:R1

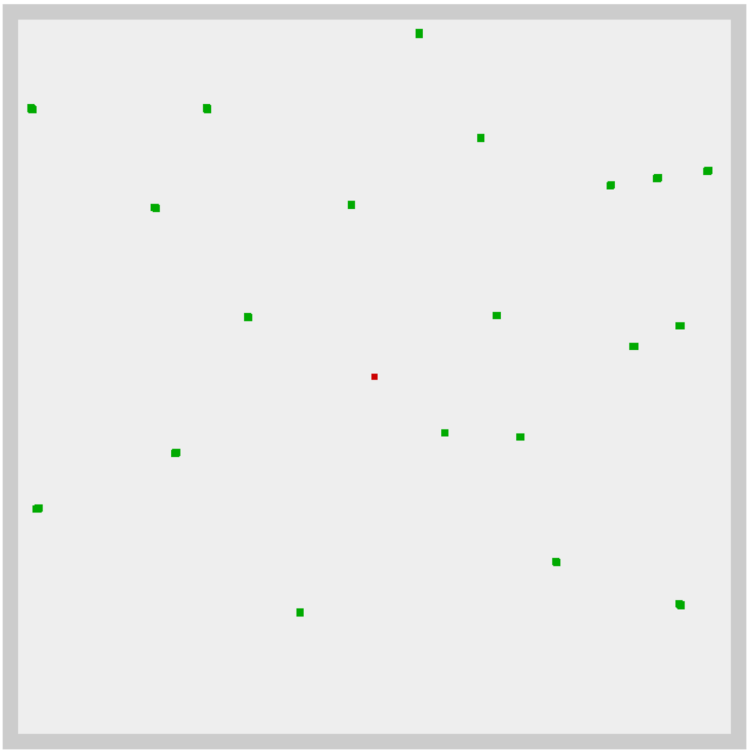

Figure 5.8: Scenario: Predator:C - Prey:R2

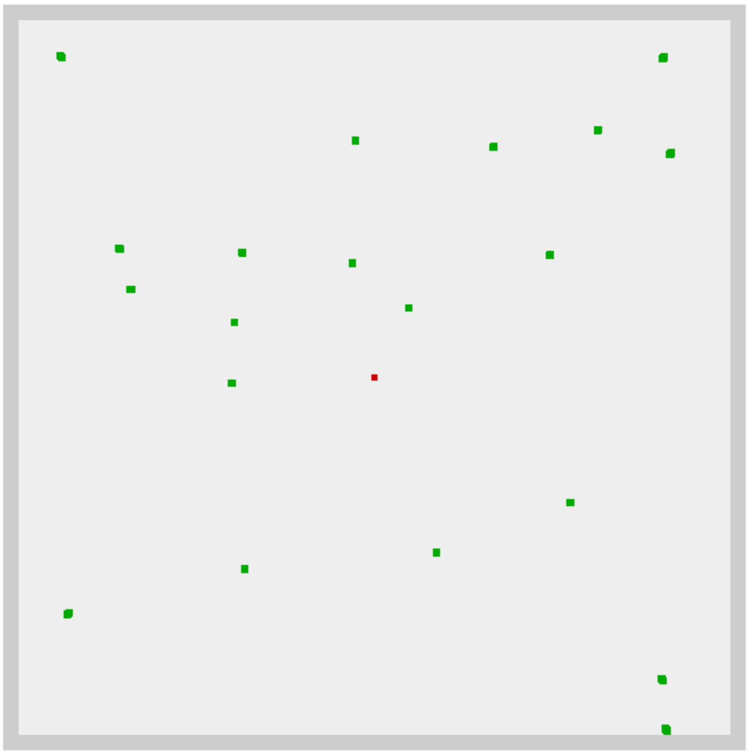

Figure 5.9: Scenario: Predator:C - Prey: R3

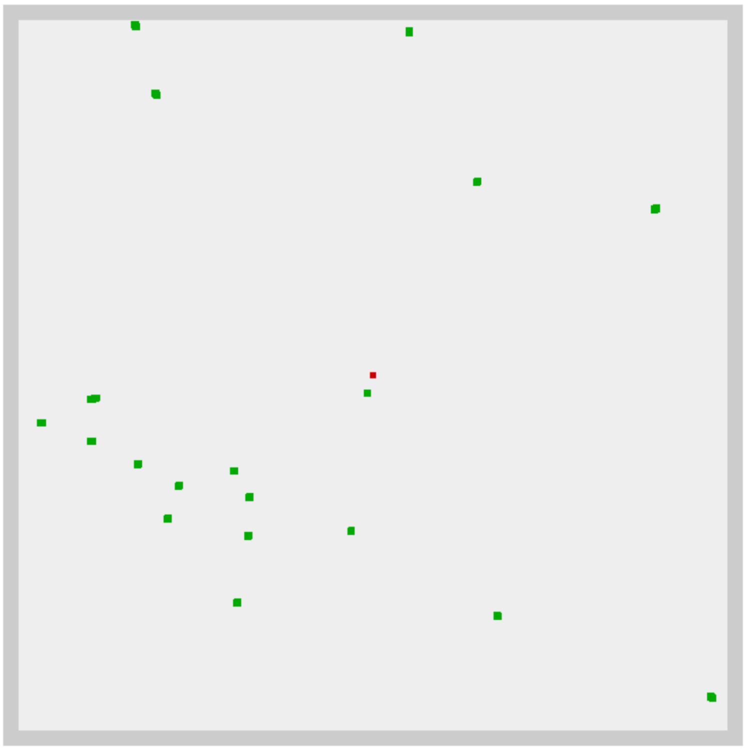

Figure 5.10: Scenario: Predator:C - Prey:R4