university of groningen

faculty of science and engineering

Emergent Symbiotic Speciation: Growing Hologenomes with Cooperative Coevolution

Ivo E. Steegstra

university of groningen / faculty of science and engineering

University of Groningen

Emergent Symbiotic Speciation: Growing Hologenomes with Cooperative Coevolution

Master's Thesis

To fulfill the requirements for the degree of
Master of Science in Artificial Intelligence
at University of Groningen under the supervision of
Prof. dr. D. Grossi (Artificial Intelligence, University of Groningen)
and
Dr. H. de Weerd (Artificial Intelligence, University of Groningen)

Ivo E. Steegstra (s2560054)

April 26, 2024

Contents

Acknowledgments

It has been quite a journey creating this thesis, and fitting with the central theme of the work, I could not have done it alone.
First I would like to express my sincerest gratitude to Prof. Dr. Davide Grossi for his supervision, wisdom and advise whenever I needed it. The freedom you have given me in following my passion has been both a blessing and a curse, and an invaluable gift. I am extremely grateful to Abe Brandsma, who has endured endless conversations with limitless patience, and has provided me with reflection and feedback from start to finish. I can always count on your counsel. I am deeply indebted to my parents, Aernout Steegstra & Ageeth Huizenga, who have always supported me and believed in me, especially in difficult times. I owe this thesis to you. A special thanks to Floor, for your love and support and giving me the motivation to keep going. You are always there when I need you. Thanks should also go to Kas Jansma and Hendrik de Boer, who have both inspired me and at the same time kept me grounded, I derive the greatest pleasure from our conversations. I would be remiss in not mentioning Bart Buitenhuis, who provided the necessary distractions. I value our friendship deeply.
Finally I would like to extend my love and gratitude to all friends, family and fellow students that have been here with me these last years. I am very fortunate to have so many kind, inspiring, intelligent, wise and fun people in my life.

List of Figures

List of Tables

Abstract

In nature, organisms larger than a microbe typically comprise organisms of multiple species. Species speciate, combine and coevolve to perform different functional roles in an ecosystem. This thesis applies this biological framework to the field of computational evolution. We introduce a model featuring several novel mechanisms applied to the framework of cooperative coevolution, and it is shown that the model is capable of producing a composite solution of multiple emergent symbiotic species from a single gene pool, without prior configuration of the composition. Speciation within a single gene pool is achieved through a novel assortative mating strategy. Emergent species are identified with unsupervised clustering, and solution compositions are formed by determining species type combinations with a genetic algorithm, thereby also facilitating meaningful coevolution of species. Finally a novel credit assignment strategy ensures survival of efficient altruism. The model is evaluated with the behavioural dynamics produced by three distinct domain models: a predator-prey simulation, function optimization, and a novel 'Toxin' model designed to showcase the game theory dynamics inherent to the model. Results show that dynamic emergent symbiotic speciation is achieved and produces good composite solutions. Species are not shown to be optimally decomposed subcomponents of the problem.

1 Introduction

Evolution is a powerful principle. It has shaped all life that we know, it operates on complex systems from language to the economy, and we can use it as a tool for creative problem solving. To do so, all we need is to define the problem to solve, the format of an evolutionary unit or 'shape' that can attempt to solve the problem and an evaluation method to judge the performance of the candidate solution. Then, subject the shape to the basic evolutionary mechanics of selection and variation, and over time the shape will form to be a solution to the problem. It is not certain that the resulting shape will be an optimal solution, but the outcome is invariably creative and requires no external designer.
The field of evolutionary computation has embraced the insight of evolution as a tool, leading to a diverse array of algorithms and mechanisms inspired by biological processes [1]. This thesis is predicated on the belief that biology still has much to offer, particularly in understanding evolutionary strategies. It endeavors to merge a fundamental concept from biological evolution into a computational model. Specifically, it explores the hologenome theory of evolution[2, 3, 4] - the principle that interspecies cooperation and coevolution are pivotal in evolution, exemplified by the fact that complex organisms typically comprise multiple symbiotic species.

One major obstacle in the process of evolution is that to evolve to a good solution, there needs to be an *evolutionary pathway* towards that solution. Every change to the shape needs to have an incremental benefit. The human species cannot grow wings over generations because the end result of flying is beneficial; every evolutionary step towards those wings needs to confer some advantage.
When the defined problem is a very complex problem, it would require a very complex solution shape. The more complex the solution shape has to be, the likelihood of finding a viable evolutionary pathway diminishes. It is unlikely for proto life to grow into a mouse, even if it is artificially subjected to the exact evolutionary pressures of a mouse. The same is true for tabula rasa agents that are given a highly complex task to perform. A designer can assist in growing the shape by introducing the problem incrementally, letting the shape first evolve to solve a simplified version of the problem, then guiding the shape step by step towards ever more complex versions. This requires a designer to know how to simplify the problem, and to create multiple versions of the problem of increasing complexity.
In nature, organisms of higher complexity are rarely isolated organisms. Every macrobe and its associates can be considered *holobionts*; complex organisms are composite organisms of many constituent organisms [5, 6, 3, 4]. The term 'holobiont' traditionally refers to a host organism and its symbiotic species. In this work a primary host is not distinguished, the collective is viewed as an integrated unit without implied hierarchy. To signify this, the term *'composite organism'* or -entity is used.
This view of complex organisms can inspire another approach: let a designer decompose the problem into smaller and more manageable subproblems, and evolve multiple shapes to solve each of the subproblems. Together these shapes form the solution to the problem. This divide-and-conquer strategy breaks up the evolutionary pathway to multiple shorter pathways that can be traversed independently. By eliminating the need for a single complex shape, both robustness (by eliminating a single point of failure) and modularity are provided. It has the advantage that the constituent shapes are able to specialize in parallel, exploring evolutionary pathways in multiple dimensions, where a single shape can only change in one dimension per iteration.
However, this approach also requires a designer to know how the problem can be broken up into multiple subproblems. Not all problems can be neatly partitioned into independent subcomponents; subcomponents are often linked together and can form an entangled web of overlapping interdependent problems, making it difficult for a designer to define what the subtasks are. Neither can subsolutions be formed in isolation of each other, the operations of one subsolution will affect the

efficacy of another. Subsolutions need to be (co)evolved together, to be adapted to each other.
In the field of cooperative coevolutionary algorithms, problem decomposition and dynamic task allocation is an ongoing area of research [7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. This thesis proposes a perspective and a method addressing this challenge, leveraging biological and game-theoretical principles to induce emergent symbiotic speciation as dynamic task allocation. In other words, growing multiple different species of individuals that perform various functions and configure to form a single complete solution, from a single population and without defining the configuration a priori. Instead of creating one many-armed creature, we create many different kinds of smaller creatures and see which work well together.

Contributions The contribution of this thesis is a general model built using existing frameworks, with the addition of three distinct novel mechanisms, one novel application of machine learning techniques to evolutionary computation, and one novel domain model.

- The first mechanic is *Genome Distance Based Mating*, an assortative mating mate selection mechanic in the evolutionary process where individuals that are selected for replication select a mate not only based on fitness but also on genomic distance. The purpose of this mechanic is to maintain solution diversity within a single genome pool, and it achieves this by mimicking *parapatric* speciation. It functions as an alternative to the *Island Model* [17].
- Unsupervised cluster detection is used to classify the emergent species in the single genome pool, and species are tracked through generations with cluster prototypes.
- The second mechanic are the *Evolutionary Compositions*. To facilitate meaningful coevolution between species and find good solution compositions of species, solutions are composed of individuals sampled according to a composition of the determined species. The solution compositions are evolved by a genetic algorithm, in parallel with the individuals population.
- The third mechanic, *Species Representation Fitness Scaling*, is a novel form of *credit assignment* - in what part did individual behaviour contribute to the collective outcome. By scaling the fitness of individuals according to the representation of their associated species type in the compositions, the mechanic couples the selection pressure of the compositions to the individuals genome population. The resulting hybrid of individual and team level selection conserves species that exhibit altruistic behaviour at personal cost.

The behaviour of the general model is examined using three domain models: a Predator-Prey model, a Function Optimization model, both extensively used in the context of Evolutionary Algorithms and Cooperative Coevolution, and a novel *Toxin model*. The Toxin model, inspired by the *Prochlorococcus* bacteria and the Black Queen Hypothesis [18, 19], is constructed to make certain dynamics of the model explicit and thus more easily observable.

Illustrative example Imagine a football coach without any experience tasked with forming a highly efficient winning team, given only the game's rules and a pool of players to choose from. Initially, these players resemble children, indiscriminately chasing the ball without any apparent strategy. Rather than assigning fixed roles or positions, the coach lets the players experiment with different strategies during practice matches. Guided by their shared motivation to win, players begin to recognize and adopt beneficial strategies while discarding ineffective ones. Encouraging competition and learning, the coach gives preference to players who frequently contribute to winning team formations.

Over time, the coach observes natural inclinations among players towards specific roles, such as offense, defense, and coordination. Simultaneously, the coach experiments with varying compositions of these evolving roles, noticing that certain combinations, e.g., a balance of defensive and offensive strategies, tend to promote team success.
Drawing from these observations, the coach facilitates the process of specialization. By consistently selecting the most proficient players for each role, the coach encourages players to hone their skills within their chosen strategies. Though a defender might never score a goal, their contributions to the team's success are no less significant.
Thus, over time, roles become increasingly specialized, and the team composition optimizes, reflecting a continuous and dynamic process of evolution. The coach, rather than dictating this process, guides it by recognizing and rewarding beneficial strategies and fostering an environment conducive to continuous learning and adaptation.

1.1 Conceptual Framework

The approach is based on two main philosophical ideas. Multilevel evolution, and emergent complexity and simplification in evolution.

Multilevel Evolution

> "*Biological function emerges from the complex organization that spans the whole scale of life, from molecules up to whole organisms or even groups of organisms.*"
>
> - Michel Morange, The Misunderstood Gene [20]

The philosophy behind the method is to enable a higher order evolution. Evolution operates on multiple levels. Dawkins emphasized the gene as the primary evolutionary unit [21]. It is the expression of the gene through ontogeny (the development of the organism through its lifespan) in the phenotype of the individual that is subject to selection. Individuals, as holobionts, are part of a symbiotic ecosystem. The different levels of life supervene on each other, and variation and selection operate on all at the same time. The theory of recursive evolution[22] postulates that evolution produces systems that are adapted to producing systems that thrive in changing conditions. Molecular genetics are such an example[23], where minor genomic alterations (stemming from RNA replication errors) can lead to potentially significant changes in the phenotype. It is a low cost high potential yield mechanism for adaptation. In light of this theory, composite organisms with distributed, though interdependent, functionality provide evolutionary pathways in as many dimensions as constituents, providing a distinct adaptation advantage over singleton organisms. Functional heterogeneity is ubiquitous in nature, as driving force of complex emergent behaviour [24].

Emergent Complexity and Simplification

> "*A central tenet [of these investigations,] was the idea that, as in our own species, the specialization of workers for specific task groups improved efficiency, and thus overall colony productivity.*"
>
> - Functional Heterogeneity in Superorganisms [24]

Another philosophical concept underlying this research is that of emergent complexity from simple parts. To produce complex behaviour, the producing systems do not need to be complex. When

growing complex behaviour through a composition of constituents, the constituents can specialize through function loss / function transfer[18, 25, 26, 27], and further streamline through reductive evolution[28]. When a subset of a population specializes, it opens up opportunity for others to do new things[19]. These new roles are then folded into the ecosystem, increasing complexity. For example the advent of agriculture in human society; when humans started farming and food production became a specialized task, it freed up others to invent new professions to progress society[26]. Further function loss and reductive evolution on specialized roles simplifies the constituents.

1.2 Research

Building upon existing frameworks in cooperative coevolutionary systems and heterogenic teams in multi-agent systems, this research introduces a novel approach to solving complex problems using emergent composite solutions. While previous methods have successfully implemented various forms of cooperative evolution, a distinguishing aspect of the contribution lies in the autonomous nature of emergent symbiotic speciation and composition forming. Unlike prior coevolutionary models, where a designer determines at least one of either the number and nature of species, or subcomponents, or the number of individuals per species in a composition, this approach allows self-organization to determine all of those.
One research objective, as is the case in much of evolutionary computation, is the concept of removing the human designer from the loop, aiming to fully entrust the design process to the algorithm itself. This shift allows for a more dynamic and adaptive system, capable of evolving solutions without predefined constraints on species composition or species population size. By doing so, the system not only becomes more efficient but also mirrors natural evolutionary processes more closely, where species and populations dynamically adjust in response to environmental pressures and opportunities. This thesis attempts to offer a unified and more organic method of evolving complex, composite solutions.

This research explores the potential of the proposed method by focusing on the primary research question:

Research Question Can evolutionary pressures within a single gene pool lead to the emergence of mutualistic species that collaboratively solve complex problems in dynamically composed teams?

To address this question, a set of hypotheses are devised, which can be found in section 5.1.

1.3 Thesis Outline

The structure of the thesis is as follows. In chapter 2 an extensive theoretical background is provided on the relevant concepts, including a primer on the fundamental evolutionary theory used throughout the thesis, concepts from biology, game theory, computational evolution and machine learning. Chapter 3 reviews the relevant literature of related research concerning cooperative coevolution, problem decomposition, niching and team heterogeneity in multi-agent systems. The methods of this research are presented in chapter 4, giving an overview of the model architecture, the three domain models, and the specifics of the component mechanics of the method. Chapter 5 features the experimental setup, in which the research question is decomposed to a set of four hypotheses, and the experiments are laid out for these hypotheses. Chapter 6 shows the results of the experiments, organized by hypothesis.

Chapter 7 discusses the results of chapter 6, provides criticisms and limitations of the work and gives further points for consideration and future work. The work is summarized and concluded in chapter 8. A list of terms can be found in the Appendix, as well as an overview of the many parameters and values, and a link to the code repository that includes results of the presented experiments and many more.

2 Theoretical Background

In this chapter some important concepts are explained that provide the context for the work. It serves as an introduction to the various disciplines and perspectives necessary to fully understand the research. It can be read as is, or used as reference material.

2.1 Biology

First, some concepts from biological evolution.

2.1.1 Evolution Fundamentals

The first concept to understand is the most fundamental, which is the principle of evolution, originally described by Darwin in his seminal work On the Origin of Species[29]. On the origin of life, much is still a mystery. At one point in the primordial soup molecules might have grouped themselves together to form self-replicators, and then into replicator systems and eventually as de novo life [30, 31, 32]. These first forms of life were able to produce more copies of themself, causing their form to stay present over time. The replication process is imperfect, and mistakes during copying take place, meaning the forms sometimes produce slightly different versions of themselves. Some changes were disadvantageous to the survival and subsequent chance of reproduction, some neutral, and some advantageous. The forms with changes that were advantageous produced, as a result of their advantage, on average more versions like themself than the others. This caused the advantageous changes, or traits, to keep existing, and disadvantageous traits to disappear. This was the first instance of Darwinian evolution; the changes of an evolutionary unit through generations, where there is selection based on traits, variability of traits and traits are hereditary.
A perspective of evolution is to see it as a force acting on evolutionary units, changing their forms with small increments over time towards a 'path of least resistance'. The path of least resistance being the path towards increasing fitness. 'Fitness' in this context means the probability of reproductive success. A higher fitness equates to a higher probability of passing on genetic material to the next generation, and a higher amount of genetic material passed on.
The fitness of an evolutionary unit is always dependent on its environment. A dolphin can survive pretty well in the ocean, but has no chance in the jungle. Say that for a given environment we can know the fitness of each imaginable shape a creature can take, represented as a numeric value, and we would plot this on a graph, that would be considered the '*fitness landscape*' of that environment. Peaks in the landscape represent 'optimal' shapes for a creature to fit its environment, and valleys shapes with low viability. The shape of a dolphin in the ocean environment would show a peak, while the same dolphin shape in the jungle environment is a valley. A peak can be considered an evolutionary *niche*. Not all niches are equal however, a niche can be considered a *local optimum* on the fitness landscape. See fig. 1 for a visualization. For a creature, any shape resulting from a mutation would be represented by one of the neighbouring shapes on this fitness landscape. Mutations will produce divergent shapes in any direction, but due to selection only the shapes with a higher fitness tend to proliferate. The gradual change of the shape of an evolutionary unit is always a neutral or upward movement on the fitness landscape. This means that any incremental change needs to be beneficial on its own – a human can't suddenly grow wings, wings can only grow if each step towards wings is beneficial in its own right. A path from one shape to another is called an '*evolutionary pathway*'. Over time, evolutionary units' shape changes ascending on the fitness landscape. Once the peaks are reached and there is no evolutionary pathway out of the peak, the evolutionary unit resides in

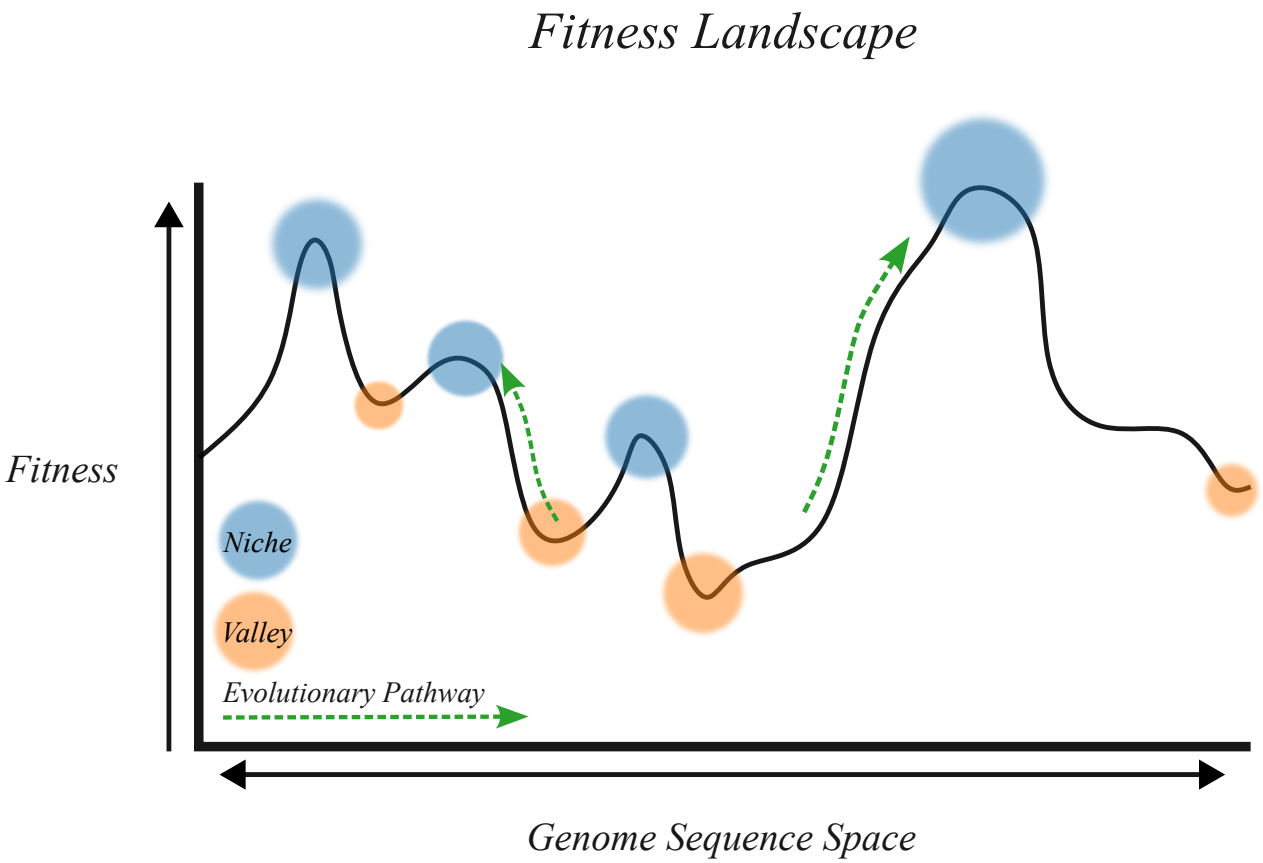


Figure 1: A two-dimensional fitness landscape. The y-axis shows the fitness, and the x-axis is the total (theoretical) sequence space: all possible geno/phenotypes. Peaks in the curve are niches, troughs are valleys. Any curve upwards is an evolutionary pathway. Note that this is a two-dimensional representation, in biological life there are *many* more dimensions.

that niche. This is called a '*punctuated equilibrium*'[33]. Also note that the shape of the peak is of concern. A lower but broader peak is more stable than a high but slim peak, seeing as that more specific requirements could be lost due to chance, which then eliminates the *quasispecies* population. A broader peak is more forgiving and allows lost mutations to reemerge [34].

However, the environment of a creature is not constant. If the environment changes, so does the fitness landscape. In a classic example of adaptive evolution, the Peppered Moth (*Biston betularia*) [35, 36] had a white colour with dark speckles, which provided camouflage when resting on the trees covered in light-coloured lichen. Then, due to the industrial revolution, trees became covered in soot and the tree trunks were blackened. Now, instead of being camouflaged on the trees, the moth stood out in its environment, causing it to be eaten by birds. The environment of the moth changed, and because of that its fitness decreased. A rare dark-coloured variation of the moth existed that in the earlier environment was at a disadvantage, and now because of the changing environment its fitness increased, causing it to proliferate. It is important to understand that the environment of an evolutionary unit consists of everything that has an effect on the entity. From physics, to the colour of the available food, to the creature's own impact on its surroundings. Evolution occurs not separate from the environment, but evolution is part of the environment.

2.1.2 Coevolution

> *"(...) because in a very real way, the ecosystem was the basic unit of life. Species creating, by their very presence, an environment for other species to work in."*
>
> - Adrian Tchaikovsky, Children of Ruin [37]

Coevolution manifests itself as a dynamic evolutionary 'dance', where changes in one species drives adaptations in others by reshaping the environmental fitness landscape. The survival and proliferation of a species depends on its ability to adapt to the changing landscape. Consider the example of gazelles and cheetahs: the gazelle needs to be faster than the cheetah or the other gazelles to survive the hunt to pass on its genes, while the cheetah needs to be faster than the slowest gazelle to catch prey and survive to pass on its genes. Each time the one creature becomes bit faster, so does the need for the other increases to become faster. This evolutionary arms race is what evolutionary biologists call the 'Red Queen' effect[38], derived from the eponymous phrase by Lewis Carroll: *"It takes all the running you can do, just to stay in the same place"* - The Red Queen, Alice in Wonderland.

In this way, evolutionary units can co-evolve. They adapt to the presence of each other over time, and change recursively to each others adaptions. Coevolution can be competitive in for example predator-prey relations, such as how the gazelle and cheetah push each other to become ever faster. Each driven by competition, the advances of the other being to the detriment of one. Coevolution can also be cooperative, where species find mutual benefit in their interactions, and evolve to exploit those interactions. For instance, flowers utilizing insects and birds as efficient pollination medium, and 'paying' the insects with flight fuel in the form of nectar. These cooperative coalitions can become very specialized, as is the case with the sword-billed hummingbird (*Ensifera ensifera*) and a flower species with a long corolla, the *Passiflora mixta*. By ensuring only one specific species of bird can reach its nectar, the flower can be sure that the bird will eat from and so pollinate only the same species of flower, which means it is efficient with its pollination. The advantage for the bird as that it has a food source that only it can reach, and so has some form of food security [39]. The same highly specialized relationship can be found in fig trees and fig wasps, where many fig tree and fig wasp species co-speciated to form exclusive pollination-food source symbiosis pairs[40, 41].

It is important to note that cooperation and competition often coexist. In the example of flowers using insects for pollination, there is competition between flowers to attract the insects. The competition leads to flowers 'advertising' to insects through colour and smell of the flower, and payments offered to the insects. Another example of mixed competition and cooperation is a symbiotic relationship between ant-acacia trees and *Pseudomyrmex* ant species[42]. The ants colonize the trees and defend them aggressively (competition based-screening). The ants attack any creature trying to eat the tree, and the tree rewards the ants with nectar from special glands. Here, the strongest, most aggressive ants will colonize the tree with the greatest nectar bounty. Stronger and more aggressive ants colonies can beat competitor ant colonies to the best producing trees, and being stronger protectors of the tree allows the tree to produce more bounty for the ants.

None of these processes are 'conscious', they happen because it is the 'path of least resistance' for evolution to take. Flowers have no eyes and ears, and are not aware that insects function as their pollination medium. They are the way they are, because the flowers that had just a bit brighter colours than their neighbours, on average, had a higher chance of producing offspring, and so the trait of being just a bit brighter persisted over time.

2.1.3 Mutualism and Symbiosis

In nature, coevolution is ubiquitous. With many different species living in the same environment, it is inevitable that symbiotic relationships form [2]. It is probable that most of the earth's biomass depend on some form of symbiotic relationship[43, 44]. *Symbiotic relationships*, meaning two different species that are closely associated, come in different forms[45]. *Mutualism* is when both species benefit by each others presence, as with the flowering plants and insects. *Commensalism* is when one species benefits, and the other is unaffected, such as when barnacles attach to whales as means of transportation. When one species benefits and the other is harmfully affected, this can be *parasitism* or *predation*[45]. Mutualistic relationships can be classified into two types: *obligate* mutualistic relationships, where species are entirely dependent on each other for survival, and *facultative* mutualistic relationships, where both derive benefits of the other but could survive without the other. Many microbes can not be cultured in isolation, they have become 'too simple' to survive on their own. They appear to function as multi-species communities [46]. Previously, mutualistic relationships between two species or two types of species are discussed. Taking a broader view, an entire ecosystem can be seen to consists of a large symbiotic relationship between many species, where the relationships are often directly or indirectly mutualistic or commensalistic. Species at different levels perform different functions of the ecosystem. A greatly simplified description – plants collect energy from the sun through photosynthesis, animals eat plants and produce manure that through bacteria and fungi becomes 'food' for plants. Plants transform carbon dioxide to oxygen, animals transform oxygen to carbon dioxide, et cetera. One can not exist without the other.

2.1.4 Speciation

The diversity of life is a product of speciation [47]. The complex system of life with its interwoven parts and interactions is a product of growth over time. Gould states that *"Speciation is responsible for almost all evolutionary change."*[48]. Speciation is when a single *species*, a collection of individuals with a certain degree of similarity, diverges and becomes two distinct species, or groups. A single group splitting to two groups is called *cladogenesis*. A single lineage changing over time is *anagenesis*. Here it is important to remember that a 'species' is a human construct. "*(...) a species, be it plant or animal, is a fiction, a mental construct without objective existence.*"[49]. It is a decision boundary people draw for which category a specific entity belongs to. There are only the entities themselves. Several processes and mechanisms have been identified to cause or contribute to speciation. The cladogenic divergence can be caused by some form of geographic physical separation:

- *allopatric* speciation separates the group with some physical barrier without any gene flow.
- *parapatric* speciation separates the group with some physical barrier with some gene flow.
- *peripatric* speciation splits and isolates a much smaller subgroup from the main.

Speciation through these methods is often accompanied by environmental change for at least one of the isolated groups, where the different fitness landscape drives the evolutionary change. *Sympatric* speciation is when a new species type develops without physical separation, but through exploitation of different available niches. Sympatric speciation is a likely outcome of resource competition and assortative mating (mating preference) [50, 3]. Assortative mating, which is a form of sexual isolation, is also a cause for low gene flow between groups. Cladogenesis through genome reduction and function loss of the Black Queen Hypothesis (see section 2.1.5) is also an example of sympatric speciation. Another theorized driver for speciation is through *Reinforcement* or the *Wallace effect*[51].

Whenever a species splits into two distinct species (cladogenesis), any hybrid offspring (child from parents of either new species) produced during secondary contact has low fitness, called *hybrid incompatibility*. This gives a reproductive benefit to the species that mate only within their new species, reinforcing the emergence of the new species. Other than clado- and anagenesis, the formation of new species also occurs through symbiogenesis, the symbiotic combination of existing species [52, 53], or integration of endosymbiotic and host cell genomes into one functional unit (see section 2.1.6). There is debate about the contemporary abundance of this mechanism [54].

2.1.5 Black Queen Hypothesis

There is a common view of evolution that it is a gradual change towards more complex systems. Evolutionary reconstructions cast doubt on this view [28]. It is true that since the origin of life, life has become vastly more complex. However, Wolf and Koonin argue in [28] that it is genome reduction that is the quantitatively dominant mode of evolution. They describe a biphasic model of evolution where short, explosive periods of complexification are interspersed by long periods of gradual simplification. The writers mention that the periods of genome reduction involve specialization contingent on environmental predictability. This rhymes with Gould's theory of punctuated equilibrium [33, 48]. The evolutionary driver for this gene loss is that if a function loses its beneficial effect, it will eventually be purged to reduce energy costs. Retaining a bigger genome is costly [19].
The Black Queen hypothesis describes an effect that concerns the evolution of dependency between organisms [18, 25, 19]. Say lifeforms of a species perform a function that is 'leaky', meaning that it is not only beneficial to itself, but others also derive benefit from it. Suppose this function is satisfying, meaning that past a certain point it is no longer beneficial to do more of it. Now if in this population a mutation occurs for an individual to not perform this function, the individual would still derive the benefit of the function, but without the cost of having to perform the function. This would give the individual a fitness advantage over the rest of the individuals in the population. This mutant will produce on average slightly more offspring than the others, and so the proportion of the mutants of this population grows. The fitness advantage is only there until the proportions have reached an equilibrium – until the proportions are such that there are enough function-having types so the function is performed just the right amount to satisfy the entire population. More accurately, once the cost and benefit of not having the function are equal. Morris in [18] suggests that there is a set of conditions for a Black Queen effect to occur:

- There is a selective advantage to becoming a beneficiary.
- The fitness advantage of losing the gene must be frequency dependent.
- The lost function is leaky.

To add to this list, in order to prevent a tragedy of the commons (catastrophic resource depletion through collective greedy behaviour, see section 2.2.4), a fourth condition needs to be met: the absolute benefit of the function must not be frequency dependent for the provider, or the producers should have preferential access to the function product [19, 55].
The result of a Black Queen effect is that it produces two new species - the providers and the beneficiaries [19]. In one sense, being a beneficiary can be seen as free-riding; they are benefiting from the actions of the other groups actions without reciprocating. The function providers are left holding the bag, or the 'black queen'. In another sense, we can see this effect as optimizing the energy expenditure of life, making it so that the function is performed efficiently. What the effect also does, is split the one life form into two distinct groups of lifeforms, where there is a one-way reliance. Within the

group of providers there is an evolutionary incentive to optimize the function, and to become better at it. The better they are at the function, the higher their fitness. The group of beneficiaries now has an energy surplus, and can 'use' it to perform a different function. They are free to develop their own niche, and perhaps perform some other leaky public goods function, that will in turn, directly or indirectly, benefit the producers. Mas et al in [19] pose that this function loss is the cause of long-lasting relationships. They find that in a system with several species in competition, if a beneficial dependency between two species emerges, their coexistence will be optimized. It is the emergence of cooperation through the emergence of dependency [19, 27, 56]. This effect has also been demonstrated in economics by Mazancourt and Schwartz in [57]. Morris and Schniter argue in [26] that Black Queen functions in an economic context provide opportunity for stable commensalistic relationships to emerge even in competitive environments, which in turn generates opportunity to develop eventual mutualistic and collective benefit.

2.1.6 Composite Organisms

> *"Symbiogenesis - The origin of organisms through the combination and unification of two or many beings, entering into symbiosis."*
>
> — K. S. Merezhkovsky, 1920 [58]

Coevolution occurs between species, and if zoomed out the entire ecosystem can be considered one large symbiotic system. When zooming in, it becomes clear that that symbiosis and mutualism also occurs within organisms. The Endosymbiotic theory[59, 60] states that eukaryotic cells originate as merger of prokaryotic organisms. The building block of eukaryotic life is a composite of multiple different (previously) lifeforms. Symbiogenesis is the process by which symbiotic partners combine to a single organism [61]. If we look at a human organism or any other macrobe, we can see that it is a microcosm of many different species. A human organism is a holobiont; a host with many different species living in and around it. Humans could not be alive without their gut bacteria.
Many microbes appear only in multi-species communities, and can not be cultured in isolation. They live in cooperative communities, and have lost the necessary functions to be self sufficient [46, 62].
The central thesis here is that species combine to become composite entities. If this effect occurs on different levels, the decision boundary of what is considered a single entity becomes arbitrary.

2.2 Game Theory

2.2.1 Fundamental Concepts

Game theory is a mathematical approach to the study of interactions between agents [63, 34]. An agent here means an entity with agency. Imagine a simple game of rock-paper-scissors between two players. Both players can choose their *strategy*: rock, paper or scissors. Rock beats scissors, scissors beats paper, paper beats rock, two like draw. If a win constitutes gaining 1 point, a draw 0 and a loss -1, we can represent a single round of the game with the *pay-off matrix* table 1. The matrix shows the outcomes for the players for each combination of actions they can take. The outcome for player A is on the left in each cell, and on the right for player B. In this matrix we can read that if player A takes action Paper, and player B takes action Rock, A receives a payoff of 1 (win), and B a payoff of -1 (loss).

Analyzing interactions with pay-off matrices provides some clarity on what rational actions would be to take, and more importantly, what trade-offs we want to make. The rock-paper-scissors game is an

Table 1: Pay-off matrix for the rock-paper-scissors game. 1: win, 0: draw, -1: lose.

		Player B ↓		
		Rock	Paper	Scissors
Player A →	Rock	0, 0	-1, 1	1, -1
	Paper	1, -1	0, 0	-1, 1
	Scissors	-1, 1	1, -1	0, 0

Table 2: Pay-off matrix for the asymmetrical Stag Hunt game. Values are relative amount of bounty.

		Hunter B ↓	
		Stag	Hare
Hunter A →	Stag	5, 5	0, 4
	Hare	3, 0	2, 4

Table 3: Pay-off matrix for the Prisoner's Dilemma. Negative values are years of prison time.

		Bart ↓	
		Cooperate	Defect
Abe →	Cooperate	-1, -1	-3, 0
	Defect	0, -3	-2. -2

example of a *zero-sum* interaction. Any gain by one player is equal to the loss for the other. The net result for both players is zero. Not all interactions are zero-sum. We will now briefly describe some important terms and concepts with classic game theory scenarios. Take the following scenario: Two hunters decide whether to hunt stag, a larger animal that can only be caught by cooperation and gives more meat, or hare, which they can hunt alone but provides less meat. Hunter B is more skilled at hunting hare than hunter A. If one hunter decides to hunt stag while the other hunts hare, that hunter will come home empty handed. If they both decide to hunt stag, they will catch one and split it for the large bounty. Since hunter B is the better hare hunter, they will catch more hare than A, and decrease the hare opportunity for A if they both hunt hare. The payoff matrix of this scenario can be seen in table 2. The matrix is both *asymmetrical* and *non zero-sum*. It is asymmetrical because both players receive different payoffs for the same actions. The point on the matrix where both hunters choose stag is the point of *maximal social welfare*, the sum of both payoffs is greatest. It is also the only *Pareto-optimal* outcome; there is no other option where all players do at least as well and one player has a higher payoff. Both stag stag and hare hare are *Nash equilibria*; there is no gain for either player by changing their strategy if the other does not also change their strategy. Neither strategy for the players is *dominant*. A strategy is dominant if it yields the highest payoff regardless of the other players' chosen strategy.
Game theory is an important abstract framework for the analysis of evolutionary dynamics, providing a mathematical system to describe interactions and dynamics [34]. What follows are some concepts from game theory that are applicable to the (co)evolution of cooperation.

2.2.2 Prisoner's Dilemma

The Prisoner's Dilemma is a classic game theoretical example illustrating a trade-off scenario between personal and collective gain. Two people, Abe and Bart, are caught at fleeing a crime scene and are interrogated separately. The police do not have enough evidence for the conviction they want, so they try to entice both parties to provide testimony against the other. Abe and Bart both have the option to either give up the other person to the police (defect), or keep silent (cooperate, with the other). If they both cooperate and stay silent, the police jails them both for a year. If one testifies against the other while the other stays silent, the testifier walks free while the silent one is jailed for three years. If they both testify against each other, both are jailed for two years. The we can see this situation represented in a payoff matrix in table 3. For both players in the Prisoner's Dilemma the dominant strategy is to defect, as defecting would lead to the highest payoff for either action of the other player. This places the Nash Equilibrium on the point defect, defect. This point is also the worst outcome in terms of social welfare, the cumulative prison time for both players is the highest here with 4 years. There is no Pareto-optimal point, and the point of maximal social welfare is when both players cooperate (2 years jail time in total).
If the Prisoner's Dilemma is played only a single round between two players, the dominant strategy is for a player to defect. If the game is played over multiple rounds, which we call the Iterated Prisoner's Dilemma, many more strategies become viable. Players can now 'punish' a defection by retaliating with a defection. A '*pure strategy*' is a definition of each action a player will take for all given situations: if a player always plays the same strategy in the same situation, we call that a pure strategy. For example, if a player always repeats the other players' last move. If players change their strategy based on probability, that is called a '*mixed strategy*'. A mixed strategy is a lottery over pure strategies, e.g., always repeating the other players' last move, but choosing a random move with a probability of 0.05. To avoid confusion, with this the meaning of 'strategy' is expanded to mean not just the current action, but the policy for deciding a move. There is no strategy that can win

the Iterated Prisoner's Dilemma [64]. A famously effective and at the same time simple strategy is Tit-For-Tat, which starts out with cooperation and from then on always plays the other players' last move. Another is Win-Stay-Lose-Shift.

2.2.3 Public goods Games

There is also a class of game theory games called public goods games. Public goods games are n-person games where players are given the option to contribute a hidden amount to the public good. One simple version is that all players contribute a hidden amount of currency in the form of tokens to a common pool. The contributions are then multiplied by a factor greater than one and less than the amount of players. The pool is then evenly divided among the players. The trade-off for the players here is as follows – anything they contribute to the pool will only be returned at a fraction. Anything they contribute, costs them personally. A player never benefits from their own contribution. But the game is not zero-sum; anything they contribute increases to total pool of wealth they collectively have. If everyone contributed all their tokens, everyone's personal wealth would be increased by the multiplication factor. Then there is the incentive of free-riding. The amount players submit is hidden, and submitted at the same time. There is no repercussion for contributing a small amount. Not contributing is always beneficial for the individual. As long as the multiplication factor is smaller than the number of players, the dominant strategy for the individual player is to contribute nothing. Any amount the player contributes, it will receive only that divided by the number of players. There is a risk of losing if the other players do not also contribute. If everyone contributed all they had, that would be best for everyone, as the net gain of value would be greatest through the multiplication factor. Here we see an important trade-off between personal and collective gain that is featured as a large theme in this thesis. There is also a version of a public goods game with a *satisfying function*. Instead of players receiving an equal part of the sum of contributions multiplied by the multiplication factor, there is a specific goal amount that needs to be reached by the players for them to receive the reward. Any contributions past the goal amount are superfluous. Think for example of an animal's warning call of an approaching danger. Making the call has an associated cost - running and hiding would be more advantageous to the individual. The function is also satisfying. One warning call might not be enough to alert the whole group, but once more than a few individuals perform the warning call the added benefit of one more diminishes sharply ([42]). In this public goods game with a satisfying function, free riding past satisfaction causes no detriment and contributes to the public good by not wasting actions. This dynamic is seen in the Black Queen Hypothesis in section 2.1.5.

2.2.4 Tragedy of the Commons

Another well-known common-pool resource dilemma is the Tragedy of the Commons[65]. In some sense, the tragedy of the commons can be seen as a negative or inverse version of the satisfying public goods game [66]. In the tragedy of the commons, which we will now represent in the same framework as a game theory game, players can take an amount from the collective pool, or commons. Then, the pool is multiplied by a factor and the players play another round. If the players collectively take as much such that the pool after multiplication is less than it originally was, the pool shrinks. If at any point the players take so much from the pool that it hits zero, the pool is gone and the 'tragedy' occurs. The real world analogy is that of farmers letter their cattle graze a commons [67]. Each extra cow a farmer can place on the commons, gains him a personal benefit, and the cost is divided among all that can use the commons. The problem is that if too many cows are placed on the commons, the commons are subject to overgrazing. The commons reduces in size, and this reduction in size is

accelerated by on the one hand having a smaller size for the multiplication to work on, and on the other hand farmers, seeing the 'inevitable' demise of the commons, placing even more cows to get the last bit of it before it is gone.
What is important is that these concepts are forces of the same things. There is a trade off between personal and collective gain or loss. In these dynamics there can be limits and equilibria. An upper limit is a satisfying function, for which contributions should not go over, and in other cases a lower limit can produce a catastrophe. These are frameworks in which we can place the research into context, how can we utilize these forces for our benefit, and be aware of the dangers. What feedback loops act for and against each other? If personal benefit is a fitness that optimizes the individual, the collective welfare is the fitness for the collective. How can we create the environment and rules such that individuals self-organize to an evolutionary stable state [68] that is Pareto-efficient and the point of maximal social welfare?

2.2.5 Evolutionary Dynamics

Instead of two players playing a single round, we consider a population of strategies encountering each other. Suppose the game played is rock paper scissors, and the population consists of an equal amount of rock, paper and scissors individuals. Assume there is an equal random chance for any two individual strategies to encounter each other, and values are large enough that the effect of a single individual are negligible. The expected value of an interaction for an individual is the average payoff of all possible interactions. For any individual with the paper strategy, it has an equal chance of encountering a rock, paper or scissors strategy. The payoffs for these interactions are 1, 0 and -1 respectively (table 1), so it has an expected interaction value of $\frac{1+0+-1}{3} = 0$. Now, lets suppose we calculate a fitness value for the individual, which we set equal to the expected value of an interaction. The proportion of individuals of a single strategy will grow or decrease according to the *replicator equation* eq. (1):

$$\dot{x}_i = x_i \left(f_i(\mathbf{x}) - \bar{f}(\mathbf{x}) \right) \tag{1}$$

Here,

- $\dot{x}_i$ is the rate of change of the i-th strategy.
- x_i the frequency of the i-th strategy.
- $\mathbf{x}$ is the population composition $\mathbf{x} = (x_1, \ldots, x_n)$.
- $f_i(\mathbf{x})$ the fitness of the i-th strategy.
- $\bar{f}(\mathbf{x})$ the average fitness of the population, where $\bar{f}(\mathbf{x}) = \sum_j x_j f_j(\mathbf{x})$.

In the initial state where $\mathbf{x} = (x_{rock}, x_{paper}, x_{scissors}) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, the population is in equilibrium – all strategies have an equal chance of encountering any other, and therefore all fitness values are equal. If there is a perturbation in the system, e.g., some papers become scissors, now rock has a higher fitness, because there is a larger chance for it to encounter scissors, and a smaller chance to encounter paper. The expected value of an interaction for rock goes up, and so does the fitness, and therefore the population of rocks will grow. It is disadvantageous to be scissors for the opposite reason, so that population will decrease. Now in the new state, it is beneficial to be paper, because there are more rocks and fewer scissors, causing that population to grow and rocks to decrease. Now it is more advantageous to be scissors, and so forth. Depending on the conditions of the system this can have two outcomes. Either the system changes with dampened oscillations towards an (not Nash) equilibrium

of equal proportions and continue to orbit that evolutionary stable state, or eventually an oscillation increases such that one of the three strategies has no individuals left, eliminating that strategy from the system. Then, with only two strategies left, one strategy will win or draw all interactions and the other loses or draws, and so the losing strategy is quickly eliminated and only a single strategy remains [69].

2.2.6 Evolution of Cooperation

The evolution of cooperation and the emergence of altruism are widely studied phenomena [70, 71, 72, 73, 74, 75, 76]. Games are often abstract versions of interactions that happen many times on many different levels, and are therefore a useful framework to study the emergence of cooperation [70]. In nature we see many examples of cooperation between members of a species, and even between species, even if it doesn't always 'make sense' at first glance. Nowak developed five rules for the emergence of cooperation in evolutionary dynamics [71].

- Kin Selection - also known as 'Hamilton's Rule', there is an evolutionary cost-benefit trade-off by providing altruistic behaviour for relatives, where your altruistic genes survive in your relatives. "I will jump into the river to save two brothers or eight cousins.".
- Direct Reciprocity - "I scratch your back you scratch mine".
- Indirect Reciprocity - "I help you, someone else helps me."
- Network (or spatial) Reciprocity - interactions are not well-mixed, we interact more with individuals in our spatial proximity.
- Group Selection - selection not only acts on individuals, but also on groups, and a group of cooperators is more successful than a group of defectors.

Group selection has been found to be a strong enabling force for altruists to emerge, even when at a selective disadvantage within their niche [73].
The evolution of a mutualistic relationship between species is also a form of evolution of cooperation. Chomicki et al. in [77] identify four pathways to mutualistic dependence:

- Dependence can arise from ecological constraints that limit a species' niche or behaviour.
- Increased mutualistic dependence can evolve via specific traits favoring or enforcing mutualistic interactions with particular species.
- Trait loss can lock a species into an obligate relationship with its partner and can relax selection on features and behaviours that have become outsourced to another species.
- Partner manipulation, in which one partner forces the other to become dependent.

The third pathway of compensated trait loss is also well known in endosymbioses. In other cases, dependence may emerge from community-level processes, such as the effects of the Black Queen Hypothesis [77, 75, 26] (see section 2.1.5).

2.3 Evolutionary Computation

Evolutionary computation translates the principles of biological evolution to computer algorithms. In biological life the fitness of a creature is determined by its ability to survive and reproduce, a measure of how well it is adapted to its environment. In the computational version of evolution, the fitness of an entity can be determined by a custom fitness function. Entities or individuals, represented as chromosomes, that better fit the custom criterion have a higher chance of surviving. High fitness solutions are recombined and mutated. In this way, the process of natural selection is leveraged to produce solutions that fit the designed criterion, instead of designing that solution. The 'creativity' of design is outsourced to evolution. Evolutionary computation has its difficulties too, it can be quite difficult to design an appropriate fitness function. There are many examples (for example see [78]) where the 'creativity' of evolution found a way to satisfy the fitness function without solving the intended problem. This is called '*reward hacking*'[79]. Evolutionary computation is often utilized in optimization problems, where the fitness function is simply the minimization or maximization of a value or set of values. However it is important to note that evolution does not produce globally optimal solutions (the single best solution possible). Solutions are competitive, and will find local optima, but deal with the same problems as any other optimization algorithm in terms of finding the global optimum. It is useful to think of genetic algorithms not as optimization process, but as adaptive systems [1]. Other challenges are, as mentioned in section 1, there needs to be a viable evolutionary pathway towards a solution for it to emerge, and the defined framework or encoding of the entity needs to have the capacity to evolve a viable solution – a single small neural network will not be able to perform all functions required for a self driving car. Evolutionary computation can also be quite slow, requiring many individual evaluations and generations for viable solutions to emerge. As such, the scalability can be poor. It can be challenging to maintain solution diversity, and balance exploitation and exploration [1, 80].

2.3.1 Genetic Algorithm

The Genetic Algorithm (GA) is a computational implementation of the evolutionary principles, first proposed by John Holland[81, 82]. It evolves a population of individuals, for which the fitness is evaluated by a constructed fitness function. The individuals ability to 'survive and reproduce' is directly tied to their ability to perform the constructed task. Those with higher fitness are more likely to be selected for reproduction. The new generation of individuals (offspring) is generated through recombination of individuals and chance mutation. Genetic algorithms, a class of algorithms based on the same principle, work by translating a problem solution to a chromosome genotype. The expression of this genotype is the phenotype. A fitness function can be applied to (or 'evaluate') the chromosome by running it in a model. Say the problem to solve is a Traveling Salesman Problem (TSP): given a list of cities, determine the visiting order of the N cities that has the shortest total travel distance. In this problem, the genotype is an ordered string of city names. The phenotype of that string is the (hypothetical) journey from city to city. In the TSP we want to minimize travel distance, so the fitness function is the calculation of the total distance traveled, being the sum of all distances between connected cities in that sequence. See fig. 2. Genetic algorithms usually work by not instantiating a single solution / chromosome / genotype, but a *population* of solutions. The whole population is evaluated and is assigned their fitness value. With these fitness values, selection can be performed on the population. Individuals that will reproduce, individuals that survive to the next generation and the individuals that are culled from the population are selected based on the fitness evaluation. There are various methods of selection with different trade offs, advantages and disadvantages. Ranking the individuals on fitness and selecting the top n individuals is called *truncation* selection. *Fitness*

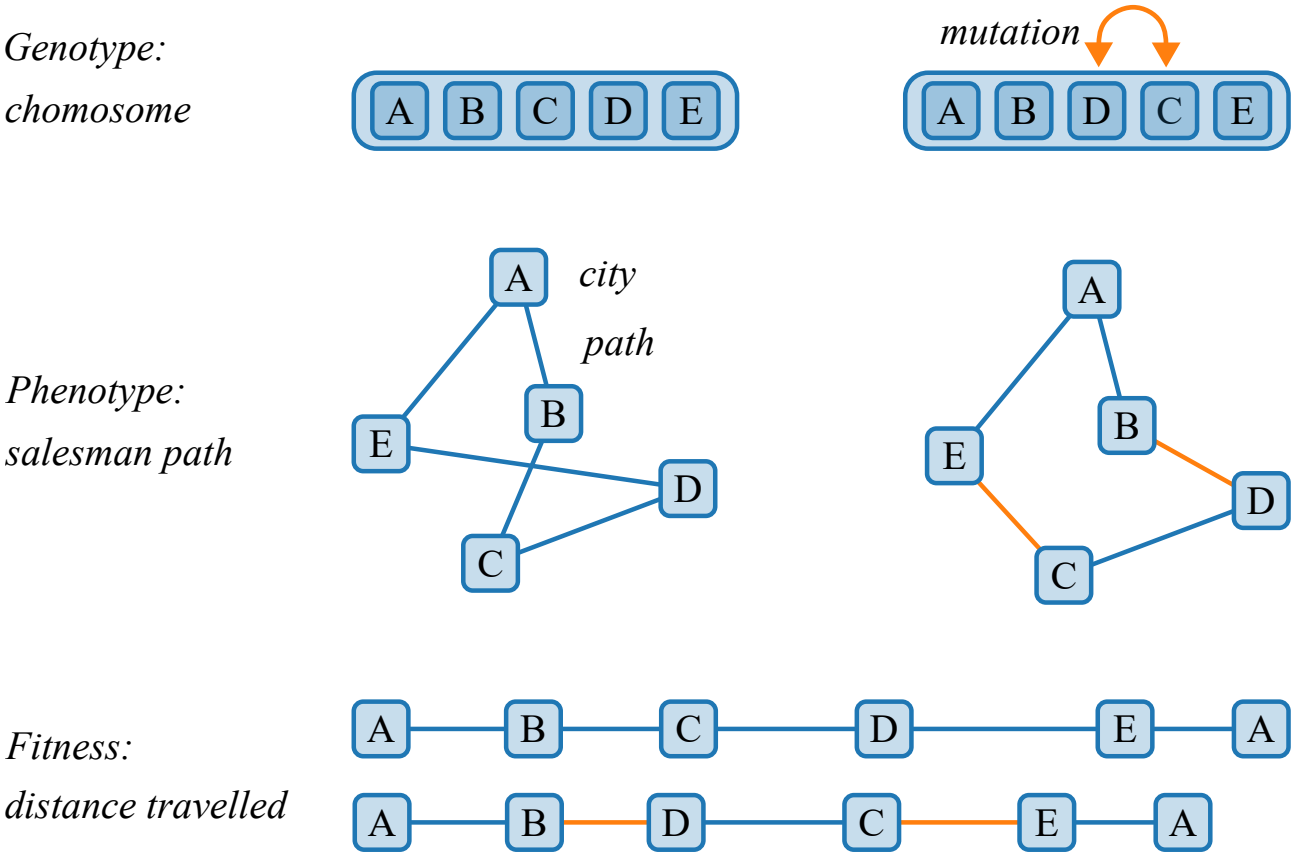


Figure 2: The traveling salesman problem solved using a genetic algorithm. A chromosome (genotype) is the order in which the cities are travelled. The phenotype is the actual path taken by the salesman. The fitness of a solution is the total distance travelled. A mutation (orange) can constitute flipping the order of two cities. In this representation, we see that the mutation produced a shorter route.

proportional or *roulette wheel* selection selects n individuals randomly, weighted by their fitness. Tournament selection selects the winners of pairwise comparisons of subsets. *Elitism* is when the top n individuals are carried over to the next generation. More detail about the trade-offs of the different used mechanics can be read in the methods section 4.3.1.

Once the subset of the population slated for reproduction is selected, they need to be paired up. Note that some genetic algorithms only mutate without recombination, or use other recombination mechanisms such as Horizontal Gene Transfer [83]. Pairing up can be done by selecting any random other from the pool of parents (with or without replacement), this can also be weighted with fitness, or it can be done sequentially to ensure all genomic material is passed on. In this research a novel method is introduced - pairing based on genomic distance: selecting the approximate nearest neighbour from a sample. More on this in the methods section 4.3.3. Once all parent pairs are selected, they are recombined to form offspring. Since chromosomes are one-dimensional strings / vectors, they can be combined using a *crossover operator*. Several crossover operators exist. A *single point* crossover operator selects a point on the genome (possibly randomly), the nth digit of the string. A child is then produced by taking all values before (and possibly including) the nth digit of parent string A, and all values after from parent string B. A second child can be created by taking the inverse operation, taking all first n values from parent B and the last from parent A. A random crossover (or *uniform* crossover) selects which gene to take from parent A or B with equal probability, per gene. Other

crossover operators exist that deal with problems such as when genes are discrete nominal values and the set of genes can not change (no duplication or gene loss), as is true in the TSP. After the first method variation is done through recombination, resulting in a set of offspring chromosomes, variation can be applied in the form of mutations. For each gene there is a small chance to change, or *mutate*. What a mutation is depends on the type of chromosome and genes. If a gene is a numerical floating point value, a mutation can be done by adjusting the value of a single gene up or down. By how much can be a set amount, a random value on a range, or a random value from a distribution. If a gene is a nominal value, a mutation can change the gene to have another value picked from the available (distribution) of genes. In the case of the TSP where a chromosome represents an ordering, a mutation can be swapping two adjacent genes.

2.4 Machine Learning Concepts

While evolutionary computation can be considered a form of machine learning [84], some other machine learning concepts that can not be categorized as evolutionary computation are used in this thesis.

2.4.1 Feed Forward Neural Network

A feed forward neural network is a function that produces an output for an input. More specifically, it is a function that can be trained to produce, or more rather approximate, certain output for certain input. Training involves shaping and reshaping the function such that it produces the desired outputs for the inputs that we give it, by adjusting the function parameters based on how well it performs. The function can be anything, from a simple plus operator to distinguishing pictures of bees from pictures of threes. A feed forward neural network is a framework that is a 'universal function approximator'. How well it does this and how complex the function can be depends on the architecture and capacity of the network. There are many complex versions and configurations, all of which will not be discussed, as only a small and simple version is used here. A feed forward neural network consists of a graph of nodes and edges, configured in a way that a node is part of a subgroup of nodes called a layer, and the edges always only connect nodes from one layer to the next. Layers are ordered sequentially. The graph is fully connected, meaning all nodes of one layer are connected to all nodes of the next layer. The first layer is called the *input layer*, the last layer is the *output layer*, and the layers in between are '*hidden*' layers. See fig. 3. Additionally, each layer has a single extra node connecting to all nodes in the next layer, called the *bias*. With each edge a value or 'weight' is associated. Each node is a temporary 'container' for a value, which is called the *activation* value, and each layer has an associated 'activation function'. An activation function is a function that maps the input value to an output value on a desirable range and distribution. There are many different activation functions for different purposes.
The network produces an output for an input in the following way. First, the input is passed to the nodes of the input layer. The number of inputs must therefore correspond with the number of input nodes. The input is numerical. These input values are going to be passed through the network, and operations are performed on them until they are passed out of the output nodes. How this happens is that from each layer, starting with the input layer, the values are passed to the nodes of the next layer. The *activation* value for each node in the next layer is calculated by taking the product of the activation value and the weight of the connecting edge of all connecting nodes, summing them, adding the bias value associated with that layer, and then applying the activation function of that layer. After this is done for each node in each layer, we have the activation values in the output nodes, which are

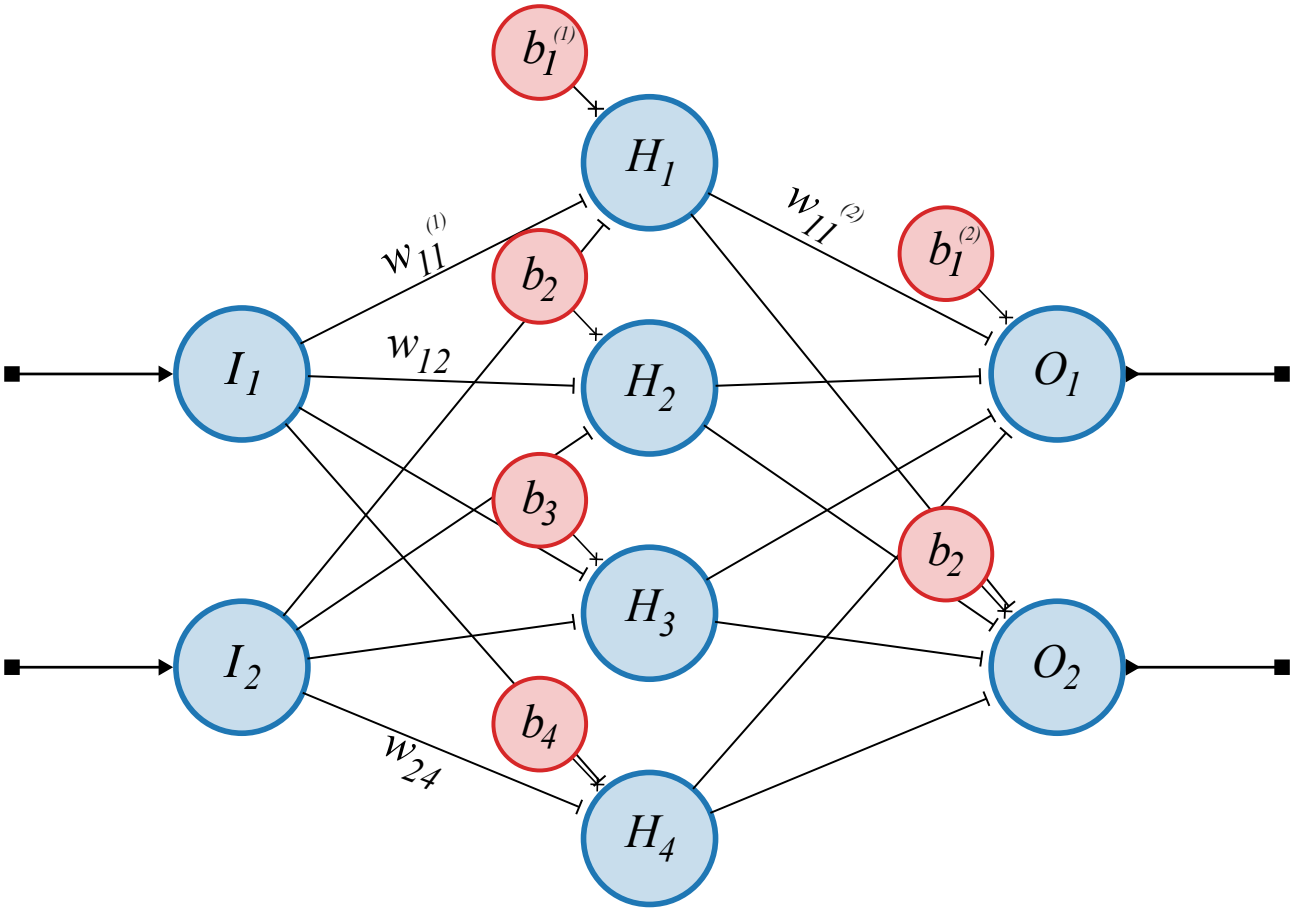


Figure 3: A feed forward neural network. Neurons are blue, biases are red. This fully-connected FFNN has two input nodes (I), one hidden layer of four nodes (H), two output nodes (O) and biases for each node in the hidden and output layer. The edges signify the connecting weights. It takes two inputs, and produces two outputs.

the output of the network. In mathematical terms, given a neural network, the value of a node i in the k^{th} layer, $S_i^{(k)}$, is determined by the following equation:

$$S_i^{(k)} = g\left(\sum_j w_{ij}^{(k)} S_j^{(k-1)} + b_i^{(k)}\right) \tag{2}$$

Here,

- $S_i^{(k)}$ is the activation value of the i^{th} neuron/node in the k^{th} layer.
- g is an activation function applied to the weighted sum of inputs.
- $w_{ij}^{(k)}$ is the edge weight from the j^{th} node in the $(k-1)^{th}$ layer to the i^{th} node in the k^{th} layer.
- $S_j^{(k-1)}$ is the value of the j^{th} node in the $(k-1)^{th}$ layer.
- $b_i^{(k)}$ is the bias for the i^{th} node in the k^{th} layer.

Note that the sum runs over all nodes j in the previous layer $(k-1)$.

In fig. 4 a cross-section of a single neuron is shown. At first, the values of the weights and biases are initialized with some initialization strategy, such as setting them all to some random value on the range [0,1]. Initially the output of the network can be anything. 'Training' the network means giving the network some input, comparing the output with the target output and adjusting the values for the weights and biases so that the output is slightly more desirable. This is done until the performance of the network is as desired or has hit a plateau (oversimplified). There are different strategies for how the weights and biases are changed. One of the most well known is by using *backpropagation*. In this research a slightly more unconventional method based on evolutionary principles is used. The weights and biases of the network are mapped from a chromosome (see fig. 5). More detail can be found in section 4.3.1.

2.4.2 Unsupervised Clustering

A feed forward neural network is an example of *supervised learning*, where for an input there is a known desired output. In this research we're interested in producing different species. It is useful to have a way to recognize when the data has shaped itself into different distinct categories. If the categories are not obvious from the data, more advanced methods are required. Clustering is the practice of grouping data into distinct categories. If the categories are known beforehand, classification can be done by creating a pattern and a decision boundary to match individual data points to a category. If the categories are not known beforehand, as is the case for emergent species, clusters of like data can be recognized using unsupervised clustering algorithms. Unsupervised clustering attempts to find patterns in data that can indicate distinct categories within the data. There are many unsupervised clustering algorithms for vector data. A simplified explanation is that the data is mapped to an N dimensional space, N being the length of the vector. Then, for each vector, the euclidean distance to other vectors is calculated, and we assign the same category to vectors in close euclidean proximity. There are methods that use centroids, which calculate the hypothetical mean of all cluster constituents, and assign each individual vector to the cluster of the centroid that is closest. The disadvantage of this approach is the assumption that clusters are roughly spherical. Other methods are density-based,

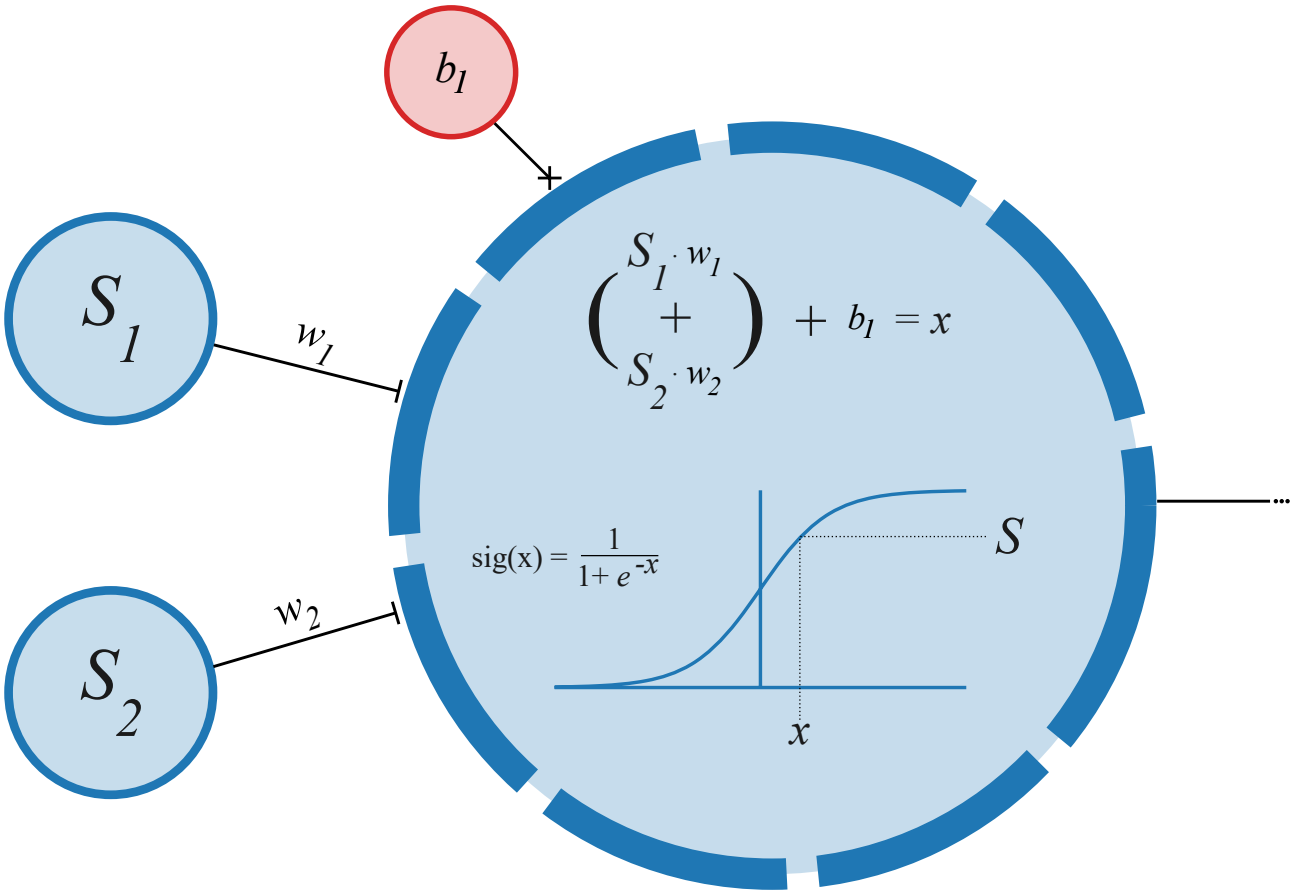


Figure 4: A neuron. This is a neuron in the hidden layer, input and output neurons work similarly. The input of the neuron are the activation values S_1 and S_2, associated weights w_1 and w_2 and a bias b_1. The output is its own activation value S. The product of the activation values and weights are summed with the bias. The result is fed through the activation function, here a sigmoid function. The result of that function is the output activation value of the neuron S.

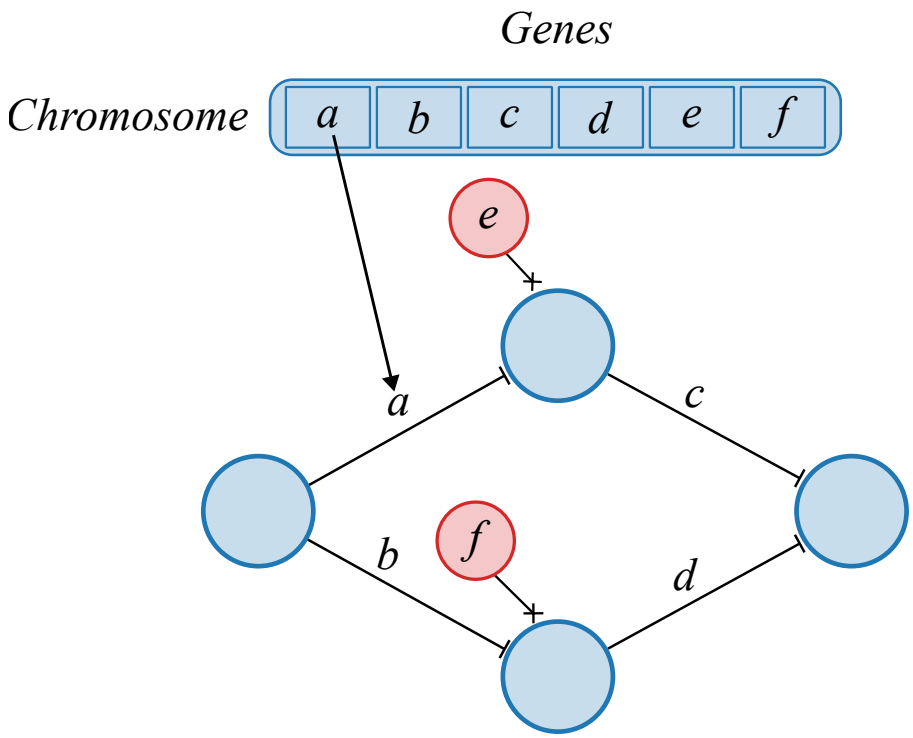


Figure 5: A neural network is initialized with the values of a chromosome. The values of genes are mapped to the weights and biases of the neural network.

that use direct distance between two points and so 'grow' a cluster. In this research, the DBSCAN [85, 86] algorithm is used. A detailed explanation and reasoning for using this technique is provides in the methods section 4.3.4.

3 Literature Review

In this chapter, first some background and general information about Cooperative Coevolution (CC) is given, followed by a comparison to Evolutionary Multi-Agent Systems (EMAS). Then, one of its main difficulties of problem decomposition is described. After that follows a short study of different approaches in earlier work that address the same problems the novel mechanics of this thesis attempt to solve. Finally, some gaps in the field are identified.

3.1 Cooperative Coevolution

Cooperative Co-Evolutionary Algorithms (CCEA) are algorithms inspired by the ecological phenomenon of mutualism, where two or more species of organisms mutually benefit from the interactions with the other, and coevolve to exploit that relationship more [7]. The main idea is to solve problems using a *divide and conquer* strategy, where a problem is decomposed into multiple subproblems, or subcomponents. Then for each subcomponent a solution is evolved. Solutions are evolved parallel, and in combination with each other, so solutions are coevolved such that they not only contribute to solving their subproblem, but also work well with the other solutions to solve the complete problem. In this way, individual solutions can be kept smaller, and much of the optimization can be done in parallel. The full solution is a composite of the subcomponent solutions. Important problems in CCEAs are that of *problem decomposition* - how can a problem be decomposed into subcomponents that are *separable*, *collaborator selection*, *individual fitness evaluation*, and *subproblem resource allocation* [7].
An illustrative example with the *CCGA-1* algorithm (code 1) [87]. Suppose a function optimization problem, where function f takes a vector v of length N as input. For a traditional genetic algorithm, to find v^* that produces the minimal (or maximal value) is an N-dimensional search problem. A population of chromosomes (vectors) of length N is maintained and subjected to evolutionary pressures. The fitness of chromosome vector v_{ci} is equal to $f(v_{ci})$. A decomposition strategy could be to break up the input vector to N subcomponents - one for each variable. Now, using a CCEA, instead of a single population with chromosomes of length N, N subpopulations with chromosomes of length 1 are maintained. The fitness of each chromosome is evaluated by combining it with (the current best) individuals from each other subpopulation to form an input vector v_{cij} for the function f. If we assume each variable is fully separable (see section 3.3), the N-dimensional search problem is now transformed to N 1-dimensional search problems, reducing the search space [87]. For clarity of the thesis, note that in this example a 'species' is the subpopulation type of one variable. What a 'species' is will vary with problem and domain. If a problem is decomposed differently and variables need to be grouped, a species type can be any grouping of variables, and in a multi-agent systems context a 'species' is a group of genomically similar agents.
CCEAs have four main advantages over traditional EAs [7].

- By decomposing the problem into separate subcomponents, (partial) parallelism can be applied to speed up the optimization.
- Separate subpopulations maintain good solution diversity.
- Decomposition into submodules increases robustness against failures of the modules, which increases the (re)usability in dynamic environments.
- Proper decomposition alleviates some of the '*curse of dimensionality*' - problems that arise with high-dimensional data.

Ma et al in [7] distinguish five main components of CCEAs: problem decomposition, subproblem optimizer, collaborator selection, individual fitness assignment and computing resource allocation of subproblems.

```
gen = 0
for each species s:
    pop[s][gen] = randomly initialized population
    evaluate fitness of each individual in pop[s][gen]
while termination_condition = false:
    gen += 1
    for each species s:
        select pop[s][gen] from pop[s][gen-1] based on fitness
        apply genetic operators tot pop[s][gen]
        evaluate fitness of each individual in pop[s][gen]
```

Code 1: Potter & de Jong's CCGA-1

3.1.1 History

The Genetic Algorithm (GA) by John Holland [81, 82], harnesses the principles of Darwinian evolution in a computational framework (section 2.3.1). The method was successfully applied to a range of problems. Optimization problems proved especially approachable to this technique, as they inherently have a well defined fitness measure for individual solutions, and solutions often fit the chromosome with genes that can mutate format. However, the method proves limited when problems increase in complexity, and not all domains are suited for the architecture. In 1994 Potter & de Jong proposed a general model for the coevolution of cooperating species in [87]. With this they introduced the first Cooperarative Coevolutionary Evolutionary Algorithm (CCEA). Their thesis was that in order to evolve increasingly complex structures, explicit notions of modularity need to be introduced in order to provide opportunity for complex solutions to evolve as interacting co-adapted subcomponents. The paper and method garnered much interest because of its ability to solve various kinds of optimization problems, and much faster than earlier evolutionary algorithms [7, 87, 88]. In 2000 Potter & de Jong published a paper[13] concerned with the emergence of coadapted subcomponents, or, into how many subproblems should the complex problem be decomposed. The concept of cooperative coevolution was applied to other domains, such as training and generating (deep) neural networks [89, 90, 91, 92, 93, 94, 95, 96]. Later developments returned to large scale optimization [97, 11, 8]. Symbiotic evolution is applied to multi-agent systems learning [98, 99, 100, 101, 102], others expand on this by introducing a multi-leveled approach [103]. A coevolutionary approach is explored in deep multi-agent reinforcement learning [104].

Benchmarks To evaluate the performance of CCEAs, benchmark problems are often used. For large-scale optimization problems, a suite of mathematical function has been created [105, 106]. This suite includes functions with scaleable dimensions, that are multi-model, non-uniform subcomponent sizes and overlapping subcomponents. CCEAs can also be evaluated in EMAS models like a predator-prey (persuit) or herding models, and various multirobot environments [7, 16, 107, 108].

3.2 Evolutionary Multi-Agent Systems

Another branch of Evolutionary Algorithms closely related and overlapping with CCEAs is Evolutionary Multi-Agent Systems (EMAS). An EMAS comprises multiple *agents* within an environment. An agent is an autonomous entity (inter)acting based on input from the environment (including other agents) [16, 15]. The framework offers a decentralized approach to solving dynamic and complex problems, and has proved to be an efficient tool for global, multi-modal and multi-criteria optimization [15]. EMAS come in different forms, including *parallel*, *cooperative-*, *competitive- coevolutionary*, and *immunological*-inspired multi-agent algorithms [15]. In cooperative multi-agent learning (cEMAS), two approaches in learning can be distinguished: *team learning* and *concurrent learning* [16]. With concurrent learning approaches, each agents behaviour is directed by their own learning entity. In team learning, a single learning entity learns the behaviour for a team of agents. Teams can be *homogeneous*, meaning a single agent behaviour type is employed by each agent in the team, or *heterogeneous*, where agents can develop unique behaviour types. A *hybrid* team consists of heterogeneous squads of homogeneous agents. The trade-off between the two strategies is the search space and specialized functionality, fully homogeneous having the smallest search space and least specialization, fully heterogeneous the largest and most, and hybrid anywhere between the extremes. Domains where a single agent (behaviour type) perform well are suited for homogeneous teams [109], while heterogeneity is better suited for decomposable domains, and the number of different required skills in a domain promotes heterogeneous specialization [110]. The level of selection can be on the team or individual in either type of team [111]. The key insight from this is that the level of granularity of control, learning and variation can be varied, and sliding these settings around are subject to trade-offs. Our model presents a concurrent learning approach with hybrid heterogeneous teams, with selection based on a combination of individual and team evaluation.

3.3 Problem Decomposition

One of the main challenges of CCEAs is that of problem decomposition. In order for a complex problem to be solved using a divide-and-conquer technique, we need to know if and how we can divide, or separate the problem into different sub-problems [14, 7]. If a problem is *fully separable*, each sub-problem, or decision variable, can be optimized independently to find the global optimum of an objective function. However, problems are not always fully separable. Interaction between variables is referred to as *non-separability* in continuous optimization literature, or as *epistasis*, *linkage* or *gene interaction* in the genetic algorithms literature [105, 11]. [7] defines separability as differentiated by problems being either fully separable (no interaction), fully nonseparable (every two variables interact), k-nonseparable (at the most k variables are interdependent) or additively separable (variables are interdependent within a subcomponent that is independent of all other subcomponents). A problem is fully nonseparable if every two decision variables interact with each other. If two decision variables interact with each other, their optimal values are interdependent. Interdependent variables should ideally be grouped in the same subcomponent, keeping dependencies between subcomponents to a minimum [11]. Interdependence between variables can greatly affect the performance of optimization algorithms in the continuous domain [11, 112]. Various variable grouping strategies have been proposed, including static-, random-, linkage-learning based-, overlap and hierarchical- and domain knowledge-based variable grouping [7].

For numerical function optimization problems, the number of decision variables is known beforehand, and so applying a CCEA *simply* requires grouping these variables in sensible subcomponents. In other domains such as EMAS, decision variables can be more obscured, and a subcomponent, or

partial solution, can be an agent with a specialized function. The problem is then decomposed in a set of distinct coevolved behaviours. For example in the Predator-Prey model (see section 4.2.1), the solution is open ended, and predator agents can discover any type of behaviour to catch the prey. A solution could involve multiple coadapted behaviours by the predators. It is possible that three predators evolving distinct behaviours is optimal. Imagine a triangulation from three different approach vectors. It is also possible that two distinct behaviours, of which one is carried out by two predators, is optimal. Imagine two chasers and one blocker. It could also be possible that a homogeneous solution is sufficient, with predators approaching at an angle influenced by model stochasticity, initial conditions or within-type differences resulting in sufficiently different approach vectors.

3.4 Maintaining Diversity in Evolutionary Computation

One difficulty in evolutionary computation is to maintain diversity in the population [113]. Diversity in an evolutionary algorithm is desirable because it gives a more optimal search over the search space. Loss of diversity leads to premature convergence, resulting in a sub- or local optimum. In CCEAs diversity is desirable because the different species can tackle different subcomponents of the problem, and so form a more complex solution through the combination of simpler sub-solutions. Most implementations of CCEAs employ a *forced subpopulations* approach. Population diversity is enforced by keeping subpopulations of solutions separate. If there is no crossover between populations, and individuals are selected for cooperation with individuals of the other populations, a diverse population is maintained, and the different subpopulations evolve to form their own partial solution. Evolving species separately mimics *allopatric* speciation as found in nature. Limiting species crossover is desirable as the offspring of two individuals specialized for different behaviours could result in a non-viable combination of two specialized halves. At the same time, occasional migration between populations can be beneficial in maintaining collective diversity and achieve higher fitness peaks [114, 115, 14]. A variant of the forced subpopulations approach that incorporates migration is the *island model*[17]. It places the different subpopulations on separate 'breeding islands'. Occasionally, individuals migrate from one island to another [115]. Allowing some crossover between islands allows some mixing of genetic material. This mirrors *parapatric* speciation. A problem with this approach is that the number of subpopulations, equal to the number of islands, needs to be determined by a designer. The number of subpopulations should be equal to the optimal number of specialists, which is equal to the number of subproblems the larger problem can be decomposed in, which is not always clear. It also offers no solution for how many of individuals from each species are required for an optimal composition in the final solution. Ideally, both the number of species, the roles of the species and the number of individuals per species is determined emergently [13].

3.5 Solution Composition Approaches

Determining the composition of the final solution - the number of unique species / subcomponents, and the number of instances of those species (in the case of cEMAS) is non-trivial.
Potter and de Jong (2000)[13] propose a solution that adds a new species, or isolated subpopulation, to the system when stagnation of the ecosystem is detected. If a species finds a niche functionality that contributes to the population, it will tend to exploit that niche, specializing in that niche. It will then be difficult for any newly added species to enter that niche. If a species is found to not contribute to the collective, it is destroyed.
Hara and Nagao proposed a genetic programming based technique that discovers the optimal number of groups and their compositions called *Automatically Defined Groups* (ADG)[116]. Bongard

proposed the *Legion System*[117, 118], a similar approach that places the groups under evolutionary control using a genetic algorithm.
Coupling the problem of team heterogeneity in cooperative multi-agent systems with the specific domain of cooperative coevolution, Nitschke et al developed the *Collective Neuro-Evolution* (CONE) algorithm[119], that coevolves multiple robot controllers using emergent behavioural specialization with the aim of increasing collective behaviour task performance. They achieve this by limiting crossover between different populations if the populations have similar specialization. A limitation of CONE is that the resulting teams are fully heterogeneous. Gomes et al developed *Hyb-CCEA*[120], which improves on CONE by allowing for the emergence of genetically homogeneous subteams, does not require *a priori* specification of the possible specializations, and (in contrast to CONE, which is limited to a specific neural network architecture) is compatible with any evolutionary algorithm.
Omidvar et al proposed a *differential grouping* strategy[11], an automatic way of decomposing an optimization problem into a set of smaller problems where there are few or no interdependencies between the subcomponents. By accurately grouping the decision variables, they can quantify the contribution of each subcomponent to the global fitness. They use this information to divide the computational resources, and allocate proportional resources according to contribution. This provides the advantage of greatly enhancing the optimization performance of imbalanced problems [11].
With many studies focused on finding the accurate number of species / teams / squads / subcomponents, Trunfio[10] is interested in finding the correct size for each subcomponent. His *Differential Evolution* approach is shown to effectively adapt the size of subcomponents during the CC search.
What all these works have in common is that they substantiate the idea that the optimal heterogeneity of a solution system is a domain and problem specific property. In [110] they found that this property is dependent not on the difficulty of the task, but that the optimal number of species is dependent on the number of skill sets necessary to successfully solve the task. This highlights the need for a general, dynamic and emergent approach to evolving the appropriate number of species and the overall solution composition.

3.5.1 Fitness

Fitness is the critical metric that determines the direction of evolution. The fitness function can also be used to 'help' evolution. In [101], the fitness function for the predator-prey model (predators need to learn to catch a prey in a 2D toroidal world) does not just reward catches, but also distance traveled in the direction of the prey, to promote solutions learning they must converge on the prey to catch it, and reward predators that did not catch the prey but did help with catching. Fitness can also be used as a tuning tool. For instance, *fitness sharing* is a technique that scales fitness of individuals based on their proximity to others, originally intended to reduce redundancy in populations [1]. By decreasing fitness of individuals in close genomic proximity to many others, an evolutionary pressure is instigated towards novel shapes, promoting solution diversity. Fitness of an individual chromosome can be calculated in various ways. *Individual fitness* is a direct representation of the individuals performance in the model, sometimes called *local reward*. In the Predator-Prey model, an individual fitness could be the amount of prey caught by the individual. *Collective fitness*, sometimes called *global reward*, is an alternative approach where fitness is attributed to the individual based on the performance of the collection of individuals. In the Predator-Prey model, a collective fitness could be the total number of prey caught by all predators. An individual fitness promotes competitive learning, a collective fitness promotes cooperative learning [111]. In some domain models an individual fitness function is clearly available, such as in the Predator-Prey domain. In others, it can be more difficult to determine the individuals contribution on the collective success.

Credit Assignment

> "*If our goal is to use a computational model of evolution to solve a decomposable problem by way of a collection of coadapted subcomponents, there must be a process by which credit or blame is assigned to each of the subcomponents for their role in the health of the ecosystem.*"
>
> - M.A. Potter [14]

The difficulty in determining each individuals' contribution to the collective performance is called the *credit assignment problem*. The fitness of the collective is often not simply the sum of the contribution of each individual, the system is complex and there are synergistic effects. Calculating the fitness of the individual as the average of all fitness evaluations of full solutions it was a part of ([101, 13]) is a reasonable heuristic approach that informs on viable partial solutions. However, it does not tell about the size of the contribution of the individual. In [13] they evaluate the contribution of an individual by comparing the collective fitness of solutions with them included with the collective fitness of solutions with the same subsolutions and only the individual replaced by another. A global reward shared fitness does not scale well in complex tasks as it does not give enough feedback on individual performances [121]. Or, it is difficult or impossible to compute because of the distributed nature of the system.

3.5.2 Species Detection & Tracking

To perform species detection, we utilize the fact that chromosomes are (numerical) vectors, and so are points in high-dimensional euclidean space. As such, performing unsupervised vector quantization on a population of chromosomes yields a classification of those chromosomes, which we can interpret as different species. There are many unsupervised clustering algorithms [122], all with different trade-offs. DBSCAN[85] is a density-based clustering algorithm, that has the advantage of working well with non-spherical clusters, large spatial data and noise [86]. The idea is that different species are clusters of spatially, and so genomically, similar data. Species are separated by areas of low density. There is little literature available within the field of EA where species are classified bottom-up through vector quantization or unsupervised clustering methods.
Detecting species in a single population is one thing, tracing those detected clusters over time is another. The labels assigned to clusters in a single population are arbitrary, so any new population is assigned new arbitrary labels. The problem can be compared to tracking clusters in streaming data, which is an active area of research [123, 124, 125]. Tracking the evolution of clusters can be useful for phylogenic analysis of the species.

3.6 Limitations of previous approaches

The concept of using evolution as a problem solver is well established, with extensive research dedicated to developing systems that produce optimized solutions through evolutionary computation. Coevolving cooperating species is proven to be an effective strategy to tackle complex decomposable problems. Finding efficient decomposition strategies remains a challenge. Existing methods managing speciation are often static, demand designer input, or lack generalizability across domains.

This thesis bridges the literature gap by proposing a unified method that leverages cooperative coevolution and team heterogeneity with a bottom-up dynamic approach to speciation and team composition, generalizable across domains. In order to achieve this, known techniques need to be supplemented by novel mechanisms and novel implementations of known mechanisms:

- Parapatric speciation through assortative mating.
- Species identification through unsupervised cluster detection.
- Tracking identified species through generations.
- Evaluating dynamic species in compositions that allow meaningful coevolution.
- A credit assignment method that both promotes individual optimization and optimization for the collective.

For these mechanisms the thesis provides theoretical grounding from principles observed in biological life and game theory.

The goal is to streamline a combination of known approaches to produce a unified novel method with the added functionality of dynamic speciation and composition. Let evolution dictate both the shape and size of the solution, on multiple hierarchical levels. In the bigger picture, this is a rudimentary step in providing the principle of evolution a framework to function as universal function approximator. While the current implementation serves as 'proof-of-concept', showing only modest advantages over existing algorithms and no performance advantage over state-of-the-art algorithms for domain specific problems, its contribution lies in establishing a foundational baseline for future explorations of the new design philosophy of emergent symbiotic speciation. This thesis addresses immediate computational challenges, and opens a pathway for advancements in evolutionary computation.

4 Methods

This chapter gives an overview of the framework, referred to as 'the model', used to test the research question and hypotheses (section 5.1). An overview of the pipeline is given, along with descriptions of the specific simulation environments (domain models). Components that are special mechanics are explained in more detail. An overview of the relevant model configurations and hyperparameters, with a brief explanation of their influences, is provided in appendix B.

4.1 Model Overview

The model used for testing the research question (section 1.2) and hypotheses (section 5.1) and showing that the desired effects (emergence of mutualists from a single gene pool) are possible consists of a general architecture managing the chromosome population, and a simulation environment - the domain models, where chromosomes are represented as agents in an environment to act. The architecture performs the operational and evolutionary tasks, and the simulations are used as testbeds that provide fitness data on chromosomes. This ensures that the method is general, and the tasks are specific. In theory, the simulation can be replaced by any task conform to some restrictions (it is multi-agent, a fitness value can be provided based on agent behaviour), and the architecture will allow symbiotic speciation to emerge.
The general overview of the model is as follows. There is a population of individual chromosomes, and there is a task (the domain model) which requires a subset of the population to be performed. The idea is to apply evolutionary forces over generations on the population, and also on the composition of the subset of individuals so that they shape into a population and composition that is optimal for the task. Since (most of) the individuals exist for only a single generation, the compositions are composed of types of individuals, where a type, or species, is identified based on genomic distance.
First the population is initialized. Each individual is assigned a species type based on its genome (section 4.3.4). A set of compositions is initialized based on these types. For each composition, a subset of individuals is sampled from the population with types corresponding to the composition (section 4.3.7). Then, all compositions of individuals are evaluated by applying them to the domain model (section 4.2). From the evaluations a fitness score is calculated for each individual and each composition. Using these fitness scores, an evolutionary algorithm is applied to the population of individuals and the set of compositions and through selection, recombination and mutation a new population of individuals and compositions is produced (section 4.3.1). The population of individuals are again assigned a species type, the compositions are aligned with the new set of types and the process is repeated for a set number of generations.
The distinction this approach has to other evolutionary computing lies in the main contributions of this thesis: bottom up species detection combined with evolutionary compositions, genome distance based mate selection and species representation fitness scaling credit assignment. The design of the model is such that these mechanisms can be replaced by more traditional mechanics (e.g., random compositions or forced subpopulations instead of evolutionary compositions) or simply turned off, to aid in comparative testing.

4.1.1 Model Pipeline

A general overview of the model pipeline is visualized in fig. 6. The initial chromosome population is generated according to the set hyperparameters with random uniform initialization. Chromosomes are translated to agents in the simulation environment. The agents are evaluated and a fitness value

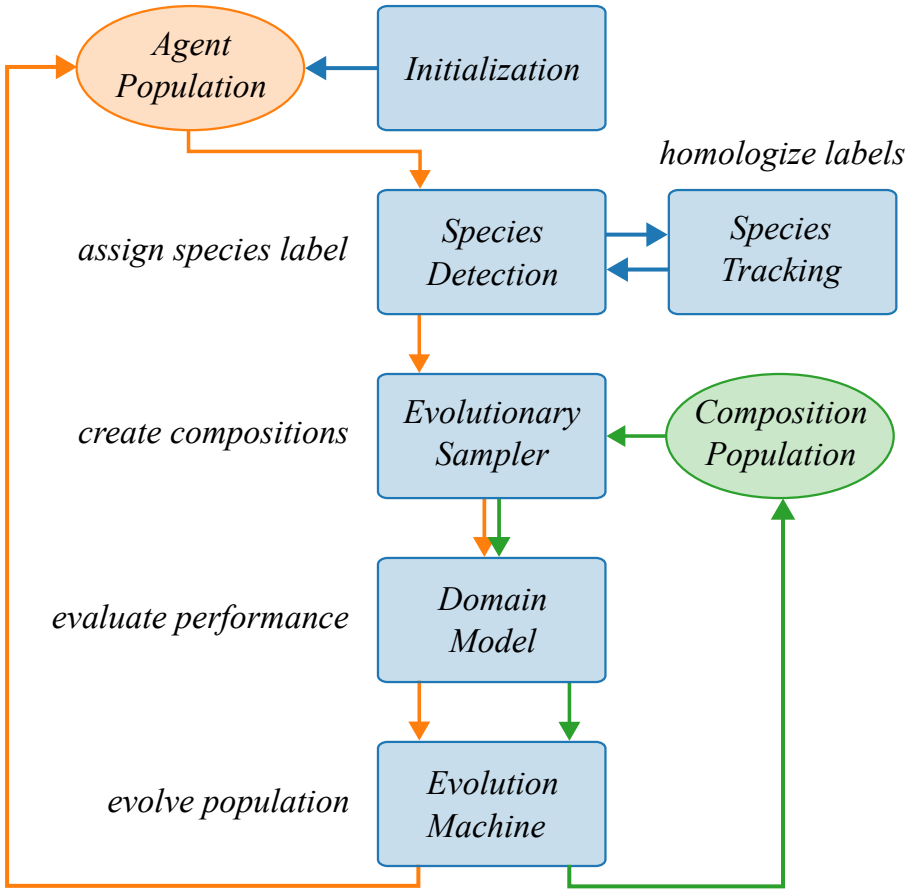


Figure 6: A general overview of the model pipeline. A chromosome population is instantiated, species labels are assigned through cluster detection, labels are homologized with species tracking, individuals are placed in compositions according to composition chromosomes, compositions and individuals are evaluated in the domain model, new generations of individuals and compositions are produced with the evaluations using evolutionary algorithms.

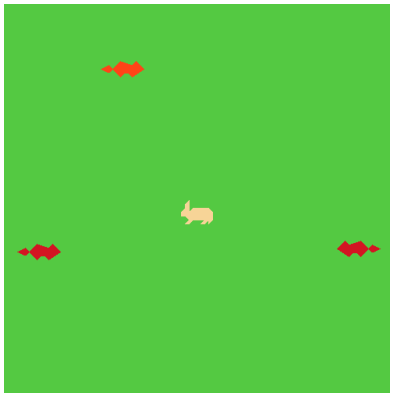
is retrieved. The chromosomes and fitness values are passed to the evolution machine, which returns the next generation of chromosomes. See section 4.3.1 for details.

4.1.2 Technologies

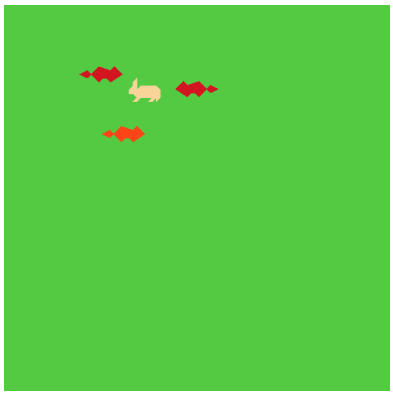
The model architecture, speciation techniques and domain models are built bespoke using the Python programming language. Some external libraries for data analysis, producing graphs and visualization are used. The complete source code can be found on GitHub, a link is provided in appendix C.

4.2 Domain Models

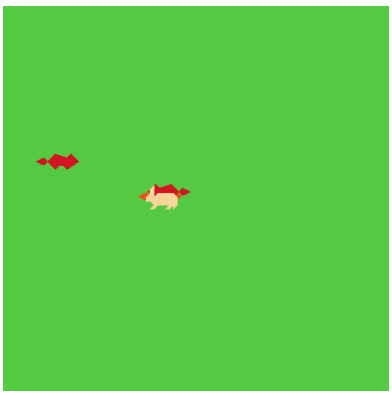
To test the symbiotic speciation architecture we have applied it to some domain models. Two common CCEA benchmark models are used, and one novel. The common benchmark domains are the Predator-Prey (persuit) model, and a model for function optimization. The novel model, called the Toxin Model, is designed to show the evolutionary game theory dynamics of the architecture. Note that the larger framework / architecture / model is also referred to as 'model'. When referring to a domain model, the specific name will always be used or it is referred to as 'domain model'.



(a) Herding prey by equidistant triangulation from far.



(b) Close proximity triangulation.



(c) Predators close in to catch the prey.

Figure 7: Visual representation of the Predator-Prey model. We see three predators (red foxes) of two different types, distinguishable by slightly different coloured fur, triangulating prey (beige bunny). The three screen captures are from different runs of the model featuring the same individuals at different stages of the capturing process.

4.2.1 Predator-Prey Model

The Predator-Prey or pursuit-evasion model is a common benchmark domain model for cooperative multi-agent systems [101, 126, 127, 16, 128, 100, 129, 130]. Among the many implementation there are subtle and less subtle differences. Here our implementation is described. In a 2-dimensional toroidal world, there is one prey (a rabbit), and three predators (foxes). It is the objective of the predator to catch the prey, by coming within a certain distance from the prey, and it is the prey its objective to evade the predators. See fig. 7 for a visualization. The predators and the prey move at the same speed. The prey its behaviour is dictated by a simple rule; it will move in the direct opposite direction of the closest predators position. This means that a single predator will never be able to catch the prey, and if all predators adopt a greedy approach of heading straight for the prey, they will never catch the prey. The predators will have to cooperate for consistent success. The behaviour of the predator is directed by a tiny feed forward neural network with two inputs - the x and y delta to the prey, a single hidden layer (of 4 nodes, biases, a sigmoid activiation function on the input and hidden layer, an identity activation function on the output layer, see fig. 3), and two outputs - an x and y forming a movement direction vector. Note that the predator can 'see' the prey and dictate their own movement, but has no direct knowledge of the other predators. The values of the weights and biases of the neural network are mapped from a one dimensional vector of 22 continuous values - the chromosome (see fig. 5).
A single run of the model consists of 150 'ticks', or steps. Predators are spawned at location (0,0), the prey spawns at a random location. Each tick, every agent determines their heading (on a continuous scale) and is moved a set distance (speed = 5) in the direction of that heading. A small amount of random noise is added to the heading of the prey. Agents are updated sequentially, first the predators, then the prey. The environment is 500 units by 500 units, agents move 5 units per tick, allowing agents to travel $5 \cdot 150 = 750$ units in a single run. The run ends if 150 ticks elapsed, or the prey is caught.

The Predator-Prey Model is an open ended search problem. The solution consists of three real valued vectors of length 22. These chromosome vectors are the genotypes of the phenotype behaviour

produced in the model. The gene values are applied to form the feed forward neural network, which determines predator action policy. The chromosome $\mathbf{c}$ is translated to a neural network (eq. (3)), the neural network, as function f taking two inputs and producing two outputs (eq. (4)), determines the action response of an agent for each input. The output, a desired x and y direction, are transformed to a heading angle (eq. (5)) in radians. This heading, the current position of the agent, and the speed parameter s are then used to calculate the new position of the agent (eq. (6)).

$$\mathbf{c}_i \rightarrow f_{\text{action}} \tag{3}$$

$$f_{\text{action}}(\Delta x, \Delta y) \mapsto (dx, dy) \tag{4}$$

$$\theta = \arctan 2(dy, dx) \tag{5}$$

$$f_{\text{move}}(x_{t-1}, y_{t-1}, \theta_t, s) \mapsto (x_t, y_t) \tag{6}$$

$$x_t = (x_{t-1} + (s \cdot \cos(\theta))) \mod 500 \tag{7}$$

$$y_t = (y_{t-1} + (s \cdot \sin(\theta))) \mod 500 \tag{8}$$

Here,

- $\mathbf{c}_i$ is the chromosome for predator agent i.
- f_{action} is the feed forward neural network as function, taking two inputs and producing two outputs.
- Δx and Δy are the difference in x and y position between the predator and prey, accounting for toroidal space.
- dx and dy are the 'desired' x and y direction for the predator.
- $\arctan 2$ is the function computing the angle in radians from origin to the dx and dy coordinates.
- θ is the desired heading angle in radians.
- f_{move} is the function that calculates the x and y coordinates for the agent for the next time step.
- x_t, y_t are the x and y coordinates for the agent on time step t.
- s is the speed parameter, giving the euclidean distance traveled in the direction θ from the current position (x, y).

Predators, or more rather their chromosome, are evaluated with the fitness function eq. (9). The fitness of predator agent i, F_i, is determined by the initial distance of the predator to the prey d_0 and the final state of the model, providing the distance of the predator to the prey at the final tick d_e and whether the prey was caught. This means an evaluation of a chromosome takes place after performing an action anywhere from 1 to 150 times. The fitness of a chromosome is partially dependent on the behaviour of the two other chromosomes that are in its team.

$$F_i = \frac{1}{10} \begin{cases} 354 + 10000 - d_e & \text{if prey is caught} \\ 354 + d_0 - d_e & \text{otherwise} \end{cases} \tag{9}$$

Here,

- F_i is the individual fitness of agent i.
- d_0 is the initial distance to the prey for agent i.
- d_e the distance to the prey at the end of the run for agent i.

354 is added to both outcomes to ensure the fitness is a positive value, 354 being the maximum possible distance to the prey (diagonal). This constraint is required for the species representation fitness scaling (see section 4.3.8). The model and fitness calculation is largely based on that of [101]. The fitness function gives some guidance to the evolution by rewarding partial solutions; predators are rewarded if they move closer to the prey.
The collective fitness CF for the Predator-Prey model is the average fitness of all predator agents, calculated with eq. (10).

$$CF = \frac{1}{3} \cdot \sum_{i=1}^{3} F_i \tag{10}$$

For a consistent assessment of the teams the stochasticity of random prey spawns needs to be factored out. For this purpose a benchmark setting is ran every couple generations. This is the same benchmark setting used in [101]. The field is divided up into a 3x3 grid, and for nine runs the prey spawns in the center of one of the 9 squares, one time in each. The benchmark score is the number of catches the team of predators make out of these 9 attempts, 0/9 meaning no catches and 9/9 a perfect score.

Justification The Predator-Prey persuit domain model is used extensively throughout the literature [101, 126, 127, 16, 128, 100, 129, 130], and so gives a solid reference point. In this thesis, it shows how the larger model framework operates on a domain model with a moderately sized open ended search space, and provides a demonstrative function by being easily visualized.

4.2.2 Toxin Model

The Toxin model is a novel model created to give an insight in various evolutionary (game theory) dynamics, and explicitly show problem decomposition. The model is roughly based on the theoretical scenario discussed in [18, 19], of simple organisms living in a toxic environment with a leaky function of detoxifying the environment. It concerns the bacterium *Prochlorococcus* living in an environment with toxic *HOOH*, for which some of the population have a gene *KatG* responsible for the production of *catalase-peroxidase*, which neutralizes the *HOOH* [131]. The idea is that the organisms perform a satisfying public goods function, and that in a saturated environment it is evolutionary beneficial to lose the function. In the model there are T distinct toxins present in the environment in a certain amount (hereafter 'level'), represented by an integer value. There are also N agents, that for each toxin either have the function to clean an amount from the environment, or do not have that function. If an agent has the function for cleaning toxin t, they are also immune to the harmful effects of that toxin (see fig. 8). There is a set cost associated with having the function, whilst not having the function has no (direct) cost. Whether an agent has this function for a toxin is determined by the bit value of a gene

on its chromosome, which is a binary vector of length T, each gene corresponding to one toxin. E.g., $a = [1,0,0,1]$, where the agent has the function active for t_1 and t_4. One model run consists of a single tick. For each toxin, for each agent with the gene corresponding to a particular toxin active, the toxin level is decreased by a set absolute amount (or until there is none of that toxin left) (see fig. 9). At the end, the fitness value attributed to each agent and its corresponding chromosome are calculated with the remaining toxin level for each toxin it is not immune to, and the cost of each function it performed. A gene cost multiplier is added to allow experimentation with different function groupings as optimal solution. A lower or greater cost for each subsequent active function should yield a different grouping of functions, which allows us to examine the effectiveness of the methods to find an optimal solution.

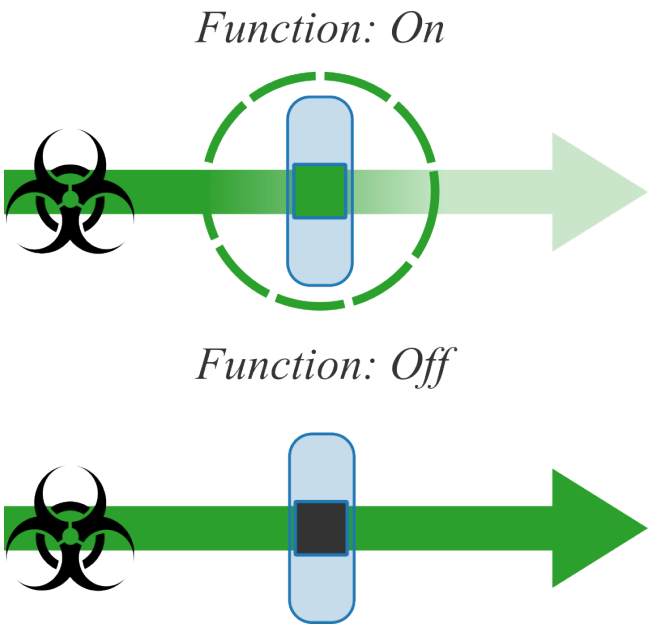


Figure 8: The cleaning function of an agent (vertical chromosome with one gene). If the gene for the specific toxin (here coloured green) is active, the agent decreases the level of toxin in the environment. The agent is also immune for the toxin, signified by the toxin coloured barrier around the agent. If the function for the toxin is off (gene coloured black), the toxin is neither protected for the toxin damage nor decreases the toxin level in the environment.

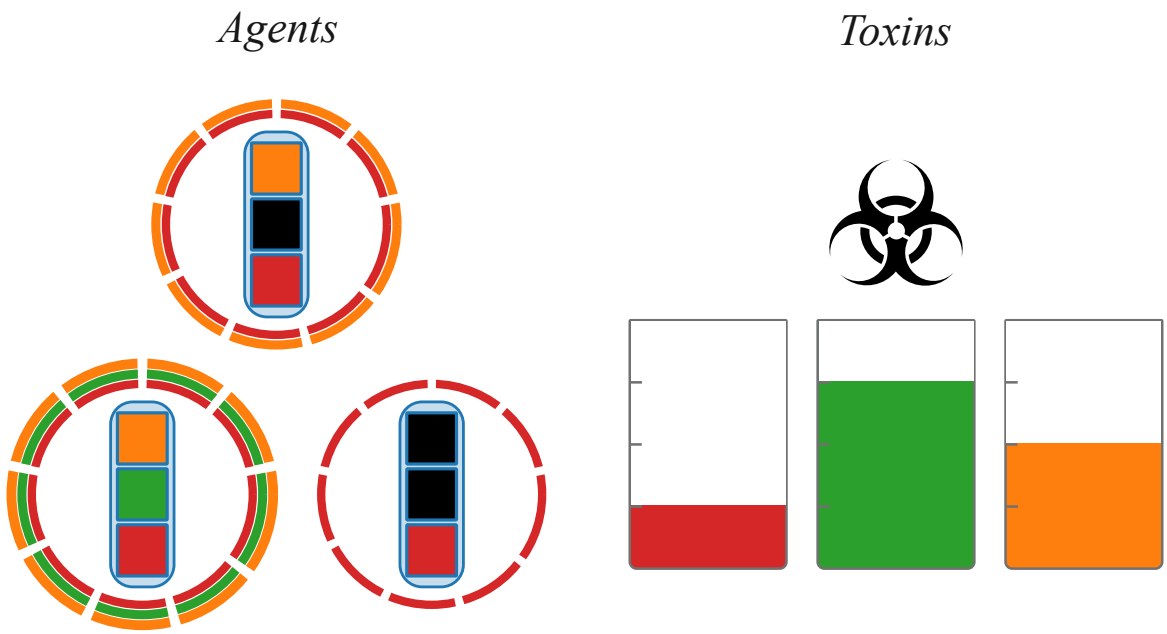


Figure 9: A visual representation of the Toxin model with three agents and three toxins. The chromosomes show the active functions of the agents for each toxin, black means off. The toxin levels decrease according to the number of agents with the corresponding function active.

The system can be described with eqs. (11) to (15) and the following variables:

$$n_i = \sum_{t=1}^{|T|} I(g_{it}) \tag{11}$$

$$c_t = c_B - \sum_{i=1}^{|A|} R * I(g_{it}) \tag{12}$$

$$GFC_i = \sum_{j=0}^{n_i-1} C_f \cdot M^j \tag{13}$$

$$TC_i = \sum_{t=1}^{|T|} c_t \cdot (1 - I(g_{it})) \tag{14}$$

$$F_i = -(GFC_i + TC_i) \tag{15}$$

Equation set 1: Toxin model equation definitions.

Given,

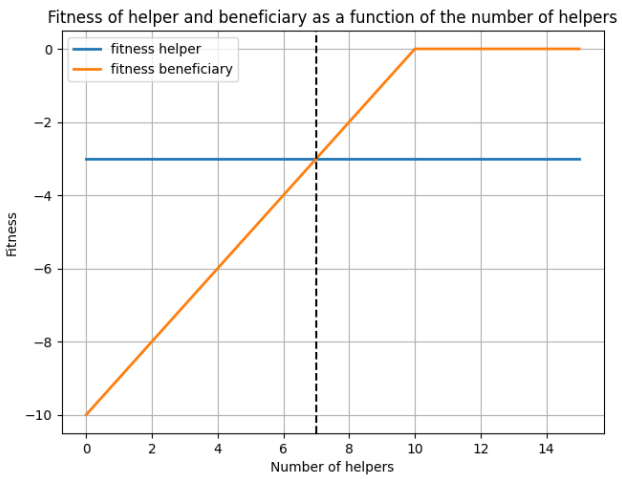
t_i	Toxin i
a_i	Agent i
T	The set of toxins
A	The set of agents
c_B	Base toxin level amount
c_t	Remaining level of toxin t
$\mathbf{g}_i$	Chromosome of agent i, a binary vector of length T
f_t	Function to clean toxin t
R	Rate of cleanup, the amount a toxin is decreased by a cleanup function
C_f	Base cost of having a gene (function) active
M	Gene cost multiplier
n_i	Total number of active genes for agent i
$I(g_{it})$	Indicator function for gene t of agent i
GFC_i	Gene Fitness Cost for agent i
TC_i	Toxin Cost for agent i
F_i	Fitness of agent i
CF	Collective fitness / global reward

Two different calculations for the global reward or collective fitness are used for the Toxin model, in different configurations. One called 'average', which is equal to the average fitness of all individuals (eq. (16)), and one called the 'toxin remainder', that calculates the total remaining toxin in the environment (eq. (17)).

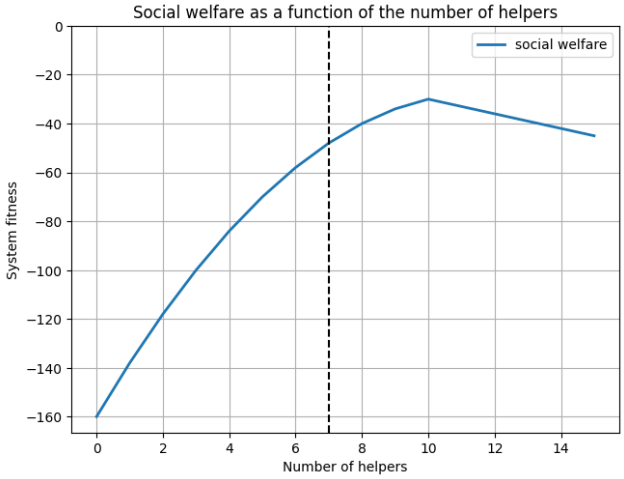
$$CF_{\text{average}} = \frac{1}{|A|} \cdot \sum_{i=1}^{|A|} F_i \tag{16}$$

$$CF_{\text{toxin remainder}} = \sum_{t=1}^{|T|} c_t \tag{17}$$

Dynamics illustration To explain the dynamics of the system (fig. 10), consider $|N| = 1$, meaning there is only a single toxin $T = \{t\}$ in the environment, $|A| = 15$, giving 15 agents, the initial toxin value $c_B = 10$, each agent performing the function to clean up $R = 1$ toxin value, and the cost for having the function active $C_f = 3$. The fitness of a 'helper' agent with the function active is constant and independent of the number of other agents with the function active. The fitness of a 'beneficiary' agent without the function active is dependent on the number of helper agents (fig. 10a). If there are 10 agents with the function active (eq. (11) $\sum_{i \in A} I(g_{it}) = 10$), the toxin level for the one toxin in the system decreases to $c_t = 0$ (eq. (12)). All 10 agents with the function active have a fitness of $F_i = -(GFC_i + TC_i) = -(3+0) = -3$ (eq. (15)), for the cost of having the function active, and the remaining agents have a fitness of 0, since they do not suffer any consequences of not having the function active. This gives the total system a fitness of -30. The Nash equilibrium of the system is when for each agent the gene cost of having the function and the toxin cost of not having the function are equal. At that point the costs of switching from active to inactive and vise versa are not worth it.



(a) The fitness of being a helper (function active) vs being a beneficiary (function inactive). The vertical line shows the payoff / fitness equilibrium. From 10 helpers, all toxin is removed and being a beneficiary incurs no cost.



(b) The social welfare of the system (the sum of all individuals' fitness). The graph shows that the best collective situation is when there are exactly enough helpers to clean all the toxin, and no more. Note that this number does not correspond to the equilibrium amount of cooperators.

Figure 10: The fitness dynamics of the Toxin model for the example parameters.

The number of agents with the function active at the equilibrium can be calculated using eq. (18):

$$N^* = \frac{(c_B - C_f)}{R} \tag{18}$$

Here,

- N^* is the Nash equilibrium number of cooperators - the number of agents with the function active for a toxin.

For the currently defined system N^* comes to 7 agents with the function active (fig. 10a). In that Nash equilibrium state if an active agent switches to inactive, the toxin level increases to 4, meaning active agents now have a fitness advantage. If an inactive agent changes to active, the toxin level decreases to 2, meaning inactive agents now have a fitness advantage. Note however that for a single inactive agent to switch to active is a cost neutral move for that agent, making the equilibrium a non-strict Nash equilibrium.
Note that the equilibrium is a Nash equilibrium that is not Pareto-optimal nor the maximal social welfare. The equilibrium is not situated at the point that the function is fully satisfied, but at the point that the costs of having the function and not having the function are equal for each individual (fig. 10b). From the equilibrium, a *Pareto improvement* is possible. If one more agent switched from beneficiary to helper, the switch would be a cost-neutral move (-3 to -3) for the individual agent, and it would bring benefit to all beneficiaries, the benefit being the size of the cleanup rate (-3 to -2). The Pareto-optimal distribution is when there are exactly enough helpers to clean all the toxin from the environment - here any change in strategy that increases the individuals payoff (switching from helper to beneficiary) reduces the payoff of all beneficiaries. In this state, the switch from beneficiary to helper decreases the individuals' payoff.

Justification Evolutionary game theory is a useful analytical framework for the study of coevolutionary models [132]. The Toxin model allows a window into the evolutionary dynamics that the speciation architecture produces. It does so through unambiguous evaluation of the fitness of the individual and system, making agent phenotype and genotype explicit, and the tune-ability of the model offers avenues for experimentation, verifying theory with practice. In the Toxin model *functions* are explicit. The benefit is that we can observe functions unambiguously. The cost is that due to their deterministic nature, functions evolve binary. A function can not be 'lost', as would be the case in a much more stochastic and complex model, such as predator behaviour in the Predator-Prey model. One mutation can remove or return a function to the genome. With parameter tuning, changing the number of toxins and the function cost multiplier, we can determine the degree of speciation required for an optimal solution. As opposed to the Predator-Prey model, where a theoretical jack-of-all-trades single genotype is able to solve the problem with three versions of itself and some stochasticty to break lockstep. Due to its origins, it is especially adept at showing the Black Queen effect. The system Nash equilibrium is a suboptimal social welfare. Species are explicit, and by tuning parameter values we can dictate what the optimal solution composition

4.2.3 Function Optimization Model

(CC)EAs have traditionally been benchmarked with a suite of optimization functions [14, 87, 105, 106]. The original paper by Potter & de Jong[87] proposed cooperative coevolution as strategy for solving high dimensional optimization problems.
An optimization problem is formally defined by a function $f : \mathbb{R}^N \to \mathbb{R}$, where the objective is to find a vector $\mathbf{x} \in \mathbb{R}^N$ that optimizes the output of f. Here, $\mathbf{x}$ is an N-dimensional decision vector, with each component x_i representing a decision variable. The function $f(\mathbf{x})$ maps this vector to a single real value, which is the objective value to be optimized.
For a minimization problem, the goal is:

$$\min_{\mathbf{x} \in \mathbb{R}^N} f(\mathbf{x}) \tag{19}$$

This means finding $\mathbf{x}$ such that $f(\mathbf{x})$ is as small as possible.
Function optimization problems can be tackled using an evolutionary computation [1, 106, 87, 10, 11, 88]. By encoding the decision variables vector $\mathbf{x}$ as a chromosome, where each decision variable is a gene, a solution has a genotype that can be evolved through variation and selection. By evaluating the chromosome with the outcome of its decision variables in the problem domain, a selection pressure to optimize is be applied. In a traditional genetic algorithm a single population of chromosomes is evolved, where each single chromosome is a complete solution to the optimization problem. In a single population where no effort is made to conserve diversity, the population will converge to a single species of solutions. In an optimization problem with N variables, each chromosome has N genes with values corresponding to the variables, making the search an N dimensional problem. In the first cooperative coevolutionary algorithm[87] the N dimensional problem is decomposed into N 1 dimensional problems, by creating N subpopulations, and so N species, of chromosomes with a single gene corresponding to the optimization variables, optimizing them separately and evolving them cooperatively. As discussed before in section 3.3, this decomposition strategy is suboptimal if the optimization problem is not fully separable and can get stuck in faux optima [7]. A solution is to group the variables that are linked in a single chromosome species, and coevolve these species. This problem is non trivial, and a benchmark suite is developed with optimization functions with various degrees of separability and overlapping subcomponents to benchmark the performance of the optimization architectures that attempt to solve the problem. While the proposed method makes

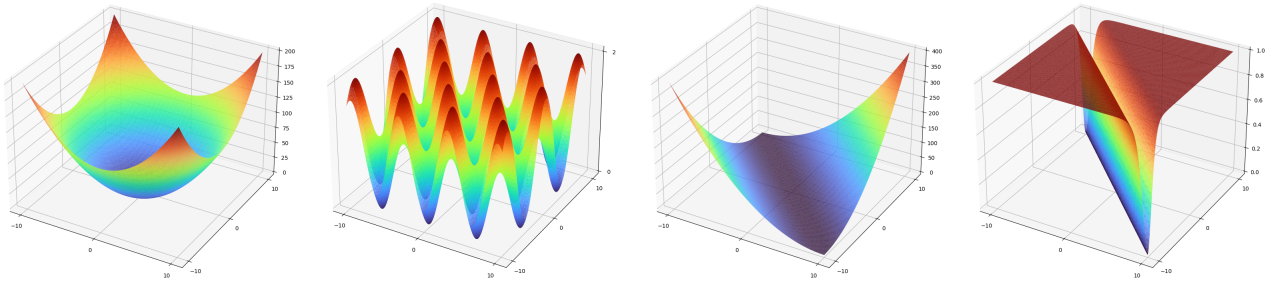
no claims in terms of optimization speed, it is still useful to see the performance and observe the behaviour on functions from known benchmarks.

The problem of variable grouping can be viewed as a hybrid team heterogeneity problem. The literature frames subcomponents as species. By viewing a specific subset of the decision vectors as a species, and we allow both the number of species in a composition and the optimization variables within a species to change, there is the capacity for an optimal decomposition to emerge. The challenge is in aiming the evolutionary pressures so that they will do so.

For an N dimensional optimization problem, an individual is a vector of length N. A full solution is the combination of (initially) N individuals, where the solution is the sum of all individuals (vector sum of the chromosomes). This initially increases, not reduces, the optimization problem from an N dimensional problem to an N^2 dimensional problem. Then, two new mutation types are introduced: the *gene knockout mutation* (section 4.3.9), and a *chromosome resizing mutation* (section 4.3.10). The knockout mutation is only applied to the individual chromosomes. It allows a mutation to occur on a chromosome that 'knocks out' a gene, by setting its value to zero and allowing no further mutations except the un-knockout mutation. Because a full solution decision vector is composed of the sum of all individuals in the composition, a gene with the value zero has no effect on the solution vector. This is an implementation detail, considering knocked out genes as removed genes has no difference. The chromosome resizing is a mutation only applied to the composition chromosomes (section 4.3.7). It allows for a mutation to occur that increases or decreases the length of the chromosome by removing a randomly selected gene or inserting a random gene to the chromosome. This in- or decreases the number of species in the solution, and so the number of subcomponents. This gives the system the capacity to evolve a solution where species are indeed vectors with a subset of variables that have linkage, and that there are the 'correct' number of species. This approach has the advantage of allowing for overlapping subcomponents, since there is no restriction for a single variable to be constrained to a single subcomponent. The slight downside to not having that restriction is that if 0 is a local minimum for a variable it has a more likely probability of ending up there and a harder time of getting out, because there is a mutation that sets the variable to 0 and entrenches disallows regular mutations to change the value, that is much more likely to occur than a regular mutation setting the value to 0.

We attempt to do so with the inspiration of the biological principles of Black Queen function transfer and genome reduction that utilizes the architectures ability of emergent speciation. By giving a fitness penalty for active genes, the intention is to create an incentive for non-linked variables to not be grouped, through selection of the quasispecies [34]. The fitness penalty for non-knocked out genes is similar to the gene cost in the Toxin model. A non-knocked out gene performs the function of contributing to the solution vector. By placing a cost on that function we place an evolutionary pressure to minimize the size of subcomponents. To counter this effect for variables that are linked there should be an evolutionary pressure for linked variables to occur in the same subcomponent.

Functions The benchmark functions used are a Sphere function eq. (20), the Griewangk function eq. (21), a custom fully nonseparable function (FNS) eq. (22) and a strongly partially separable function (SPS) eq. (23). For each function $f : \mathbb{R}^N \to \mathbb{R}$, and where $\mathbf{x} = [x_{i1}, x_{i2}, \ldots, x_N]$. For f_{SPS} $\mathbf{x}$ is divided into k subcomponents such that $\mathbf{x} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(k)}]$, where $\mathbf{x}^{(k)} = [x_{k1}, x_{k2}, \ldots, x_{km}]$.



(a) Sphere (b) Griewangk (c) FNS (d) SPS

Figure 11: 3D projection of the optimization functions with $N = 2$. These images give an oversimplified view of the dynamics, as N is usually larger than 2.

$$f_{\text{Sphere}}(\mathbf{x}) = \sum_{i=1}^{N} x_i^2 \tag{20}$$

$$f_{\text{Griewangk}}(\mathbf{x}) = \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{N} cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{21}$$

$$f_{\text{FNS}}(\mathbf{x}) = (\sum_{i=1}^{N} x_i)^2 \tag{22}$$

$$f_{\text{SPS}}(\mathbf{x}) = \sum_{i=1}^{k} g_{\text{subf}}(\mathbf{x}^{(i)}) \tag{23}$$

$$g_{\text{subf}}(\mathbf{x}^{(i)}) = tanh(\sum_{j=1}^{m} x_{ij})^2 \tag{24}$$

$$\tag{25}$$

Equation set 2: Function optimization functions.

The solution to each objective function has a standard offset applied to it using eq. (26), where $f : \mathbb{R}^N \to \mathbb{R}^N$. The offset to each value x_i is the index i of the value starting at 0 (+0 for the first, +1 for the second, +2 for the third.) This is done so that the solution to every optimization is not 0 for each variable. This is to prevent 'cheat' convergence to the optimum through gene knockout mutations, see section 4.3.9.

$$f_{\text{offset}}(\mathbf{x}) = [x_1 + 0, x_2 + 1, \ldots, x_N + (N-1)] \tag{26}$$

The functions are selected to demonstrate the behaviour of the system and capability to find optimal solutions under varying conditions of separability. The FNS function is fully nonseparable. The SPS

function is divided into subcomponents of size m, where each subcomponent is fully separable from each other subcomponent, and strongly nonsepararable within the component. The *tanh* function makes the subcomponent sensitive to small variation, but with diminishing returns for larger variation. The Sphere and Griewangk functions are commonly used as benchmark functions for function optimization.
First, the solution vector $\mathbf{x}$ without offsets is calculated by taking the element wise addition of all chromosomes in the composition with eq. (27).

$$\mathbf{x} = \sum_{i=1}^{n} \mathbf{c}_i \tag{27}$$

Here,

- $\mathbf{x}$ is the solution vector without offset applied.
- $\mathbf{c}_i$ is the i-th chromosome in the composition, an N-dimensional real valued vector.
- n is the number of chromosomes in the composition.

Then, the offsets are applied to create the solution vector $\mathbf{x}'$ with eq. (28).

$$\mathbf{x}' = f_{\text{offset}}(\mathbf{x}) \tag{28}$$

Now the objective value y is calculated by applying the solution vector with offsets $\mathbf{x}'$ to the objective function f with eq. (29).

$$y = f(\mathbf{x}') \tag{29}$$

Here, f is any of eqs. (20) to (23).
For each individual chromosome $\mathbf{c}_i$, the gene fitness cost is calculated with eq. (30).

$$GFC_i = |\{g_i \in \mathbf{c} : g_i \neq 0\}| \tag{30}$$

Here,

- GFC_i is the fitness penalty for each non-knocked out gene.
- $\mathbf{c}$ is the chromosome, an N-dimensional real valued vector.
- g_i is the i-th gene value on the chromosome.

Finally, the individual fitness of any chromosome F_i can be calculated with eq. (31).

$$F_i = -(y + GFC_i) \tag{31}$$

The collective fitness CF is equal to the objective value, eq. (32).

$$CF = f(\mathbf{x}') \tag{32}$$

Justification Function optimization problems are a common benchmark problem for cooperative coevolution [106, 105, 87, 10, 11, 88]. Applying the larger model framework to the explicit task of function optimization, specifically with functions of different levels of separability provides insight in the models ability to correctly group linked decision variables in subcomponents.

4.3 Component Mechanics

Other than the domain models, the larger framework model has many components that require elaboration. In this section the basic evolutionary mechanics are detailed, as well as the types of fitness, genome distance based mate selection, species detection and tracking, the need for agent sampling and how that is done through evolutionary compositions, the species representation fitness scaling mechanism, knockout mutation and chromosome resizing.

4.3.1 Evolutionary Mechanics

The basic evolutionary unit of the model is the chromosome. A chromosome is a vector of a length, the length corresponding to the number of genes on the chromosome. A chromosome can have one of multiple types of genes; the chromosome vector can consist of continuous values (floating points or integers) (as in the Predator-Prey model), of booleans (as in the Toxin model), or nominal names (as in the compositions, see section 4.3.7). The length of the chromosome is consistent for all chromosomes in a simulation and generally does not change over time, with the exception of chromosome resizing (see section 4.3.10, applied only to the composition chromosomes in the Function Optimization model). A generation of chromosomes is evolved to a new generation through the evolutionary operations of selection, crossover and mutation. In the model, evolutionary operations are preformed on the population using what in fig. 6 is dubbed the '*Evolution Machine*', an overview of which is given in fig. 12.

Initialization Chromosomes are initialized with random values. The initialization can be passed boundary conditions for the values, by default these are [-1,1]. Compositions are initialized randomly with values of the detected species labels of the first generation of agents.

Selection Chromosomes are ranked according to their obtained fitness value. Set parameters (see table 7) dictate the number of chromosomes to be culled from the population n_c, and the number of individuals that will be eligible for reproduction n_p. The n_c lowest ranking individuals are culled. The number of offspring to be produced n_o equals n_c. The n_p highest ranking individuals are selected for reproduction, these are the set of parents P.
From the set of n_p highest ranking individuals P a *first parent* is selected. In the model there are three types of selection processes possible: random selection, sequential selection and fitness proportional selection. Random selects a random first parent (with replacement), sequential loops through the set highest to lowest, fitness proportional selects a random individual weighted to the relative fitnesses of the individuals in the set.
For each *first parent*, a mate, or second parent is selected from P. There are three types of mate selection processes possible: random selection, fitness proportional selection, and approximate nearest neighbour selection. Random selection selects a random other from P. Fitness proportional selects a random other weighted to the relative fitnesses. Nearest neighbour selection takes a sample of P, and compares the genomic distances of the first parent to all in the sample, and selects the one with the smallest distance. Genomic distance for floating point genes are calculated using the *L1 Norm* or *Manhattan Distance*, or total absolute distance between genes. For nominal and binary genes the distance is equal to the *Hamming distance*. See section 4.3.3 for details.

Crossover The two selected parents produce offspring using random crossover. For each parent pair, two offspring are generated. For each gene, a random value determines whether the gene from

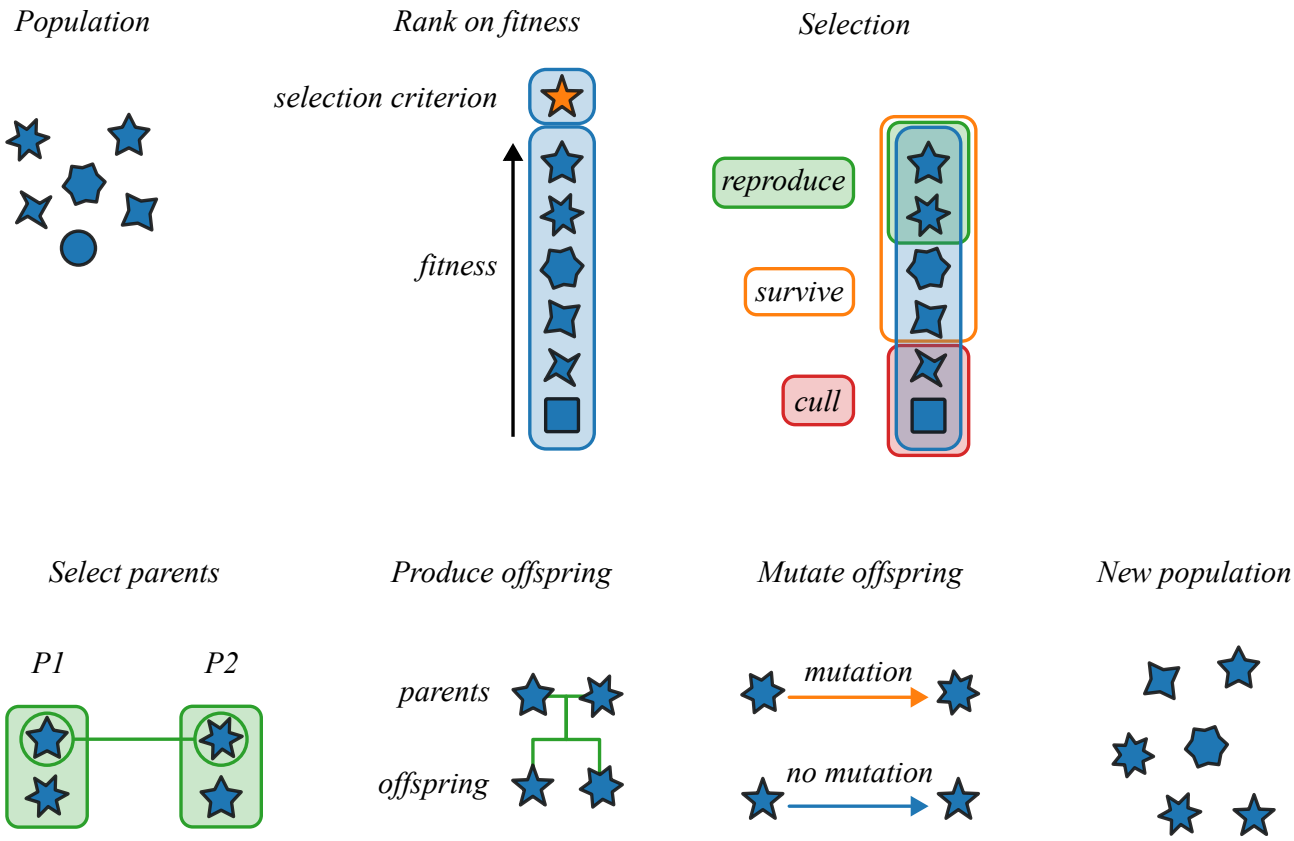


Figure 12: Schematic overview of the different steps taken to evolve a population from one generation to the next. First the individuals are ranked on their fitness towards a selection criterion. In the larger model this is the performance of the individuals and compositions in the domain models, in this schematic overview it is signified by their likeness to the star shape. Once ranked, a subset of the population is selected to produce offspring, a subset is culled from the population. The subset not selected to be culled survive to the next generation. The variables determining the sizes of these subsets are independent - e.g., the full population can be selected for reproduction and to be culled. Reproducing individuals are paired up, and produce offspring. For each offspring, there is a chance for a mutation to occur. The newly produced offspring and the survivors of the previous generation form the new generation.

parent A or parent B is inherited. The gene of the other parent goes to the other offspring. In this way, no genes are lost.

Mutation For all newly produced offspring, there is a small chance for its genes to mutate. This *mutation probability* $\in [0,1]$ applies to each gene individually. For binary genes, a mutation flips the gene to the opposite value. For nominal genes, a mutation changes the value to any of the other available gene values. For continuous genes, there is a second parameter called the *mutation range* $\in [0,1]$. To mutate a gene a delta value is applied changing the value up or down slightly. The delta value is sampled from a uniform distribution from the range [-*mutation range*, *mutation range*].

4.3.2 Fitness Calculation

Fitness determines the selection pressure. In the domain models, there are several ways the fitness of an individual can be determined. The individual fitness represents the performance of a single agent in the model. In the Predator-Prey model see eq. (9), in the Toxin model see eq. (15), in the Function Optimization model see eq. (31). The collective fitness represents the performance of the combination of agents in the domain model. In the Predator-Prey model the collective fitness calculation is the average individual fitness of all participating agents eq. (10), in the Toxin model it is either the average fitness of all agents (eq. (16)) or the toxin remainder (eq. (17)), in the Function Optimization model it is the objective value (eq. (32)). The difference is that the collective fitness is the same for each participating agent in the composition in the domain model run, and the individual fitness individually determined per agent.
The *individual fitness* allows specific tuning of behaviour or shape growth, such as introducing a penalty term for each active gene in the Function Optimization model minimizes the size of sub-components. It also allows the collective fitness to be a function of the collective personal fitnesses, such as in the Toxin model where the performance of the system is determined by the fitness of each individual.
The *collective fitness* makes sure that the evolution of the system is aimed at a collective objective - the thing that we want the system to achieve. The individual fitness promotes individual specialization and efficiency.
The *personal fitness* of an agent is the combined individual fitness and collective fitness (eq. (33)), where the relative weight is made tunable through a parameter called the 'collective fitness weight' (cfw), which in the experiments is only ever set to 0 or 1. This is used for some experiments where we want the fitness of an individual be fully determined by either the collective or individual fitness.

$$PF_i = (1 - cfw) \cdot F_i + cfw \cdot CF \tag{33}$$

Here,

- PF_i is the personal fitness of agent i.
- cfw is the collective fitness weight parameter $\in [0,1]$.
- F_i is the individual fitness of agent i.
- CF is the collective fitness.

The 'final' fitness of a chromosome is determined by taking the average of all obtained personal finesses from each model run it participated in (eq. (35)). Note that, if applied, the fitness scaling of section 4.3.8 is applied to this value, before selection.

Given that AR_i is the set of domain model runs where agent chromosome i is in the set of agents A for that model run (eq. (34)):

$$AR_i = \{r \in R | i \in A\} \tag{34}$$

$$FF_i = \frac{1}{|AR_i|} \cdot \sum_{r \in AR_i} PF_{i,r} \tag{35}$$

Here,

- AR_i is the set of domain model runs featuring the agent chromosome i.
- R is the set of model runs $\{r_1, r_2, \ldots, r_n\}$.
- A is the set of agents for model run r $\{a_1, a_2, \ldots, a_n\}$.
- FF_i is the 'final' fitness of agent chromosome i.
- $PF_{i,r}$ is the personal fitness obtained by agent chromosome i in model run r.

The fitness of a composition chromosome for a single model run is equal to the collective fitness (eq. (36)).

$$FC_i = CF \tag{36}$$

Here,

- FC_i is the fitness of composition chromosome i.
- CF is the collective fitness achieved by the agents that comprise the composition this domain model run.

The final fitness of a composition is determined by taking the average collective fitness obtained in all runs it or any other composition that is identical to it formed the agent composition (eq. (38)). Given that CR_i is the set of domain model runs where the composition of agent types C is equal to the composition chromosome i for that model run (eq. (37)):

$$CR_i = \{r \in R | C = i\} \tag{37}$$

$$FFC_i = \frac{1}{|CR_i|} \cdot \sum_{r \in CR_i} FC_{i,r} \tag{38}$$

Here,

- CR_i is the set of runs featuring composition chromosome i or a composition identical to i.
- FFC_i is the final fitness of composition chromosome i.
- $FC_{i,r}$ is the fitness obtained by a composition equal to composition chromosome i in domain model run r.

4.3.3 Genome Distance Based Mate Selection

One mechanic employed to promote speciation is that of approximate nearest neighbour mate selection. One of the problems of standard genetic algorithms is that it is difficult to maintain population diversity. Maintaining two different species in a single population with uniform interbreeding is less stable than maintaining a single species, so eventually the population will converge to a single species.

In a forced subpopulations approach this problem is solved by completely separating subpopulations from each other, allow no (or very little) gene flow between subpopulations, meaning a single species emerges per subpopulation. Since with emergent speciation the number of species and so subpopulations is variable at all times, a different approach is required.

To restrict the gene flow between species in a dynamic way, mate selection can be made dependent on genomic distance between individuals. This mimics the biological equivalent of mates selecting mates that are like them, introduces a specializing feedback loop, and allows for parapatric speciation to occur.

A chromosome can be viewed as an n-dimensional vector. The (genomic) distance between two vectors with continuous valued genes can be calculated by calculating the L1 norm or Manhattan distance between the two vectors (eq. (39)), or taking an edit distance for binary or nominal gene values.

$$D(\mathbf{a},\mathbf{b}) = \sum_{i=1}^{n} |a_i - b_i| \tag{39}$$

Here,

- $D(\mathbf{a},\mathbf{b})$ is the distance function, taking two vectors as input arguments.
- $\mathbf{a}$, $\mathbf{b}$ are vectors of length n.
- a_i, b_i are the i-th element of their respective vectors.

Calculating the closest neighbour for each first parent in high dimensional space is computationally costly, and quickly becomes unfeasible for larger populations. Instead a random sample of the eligible parent population can be taken, and from this the nearest neighbour can be selected. See fig. 13 for a visual representation, and code 2 for the pseudocode algorithm. Taking the nearest neighbour from a sample has the added benefit of *tunable* stochasticity. The larger the sample, the closer the nearest neighbour will be on average. Setting the sample value to the fraction of the population size that reproduces ensures the true nearest neighbour, and setting it to 1 gives a random eligible mate. Selecting the nearest neighbour from a random sample enables that most mating is done with a close mate, and there is still the possibility for species crossbreeding. Again, there is the trade-off between exploitation and exploration.

Alternative methods that will not be explored here are only selecting mates from the same species with a small chance of selecting a random other, or distance proportional mate selection. See section 7.4 for more details.

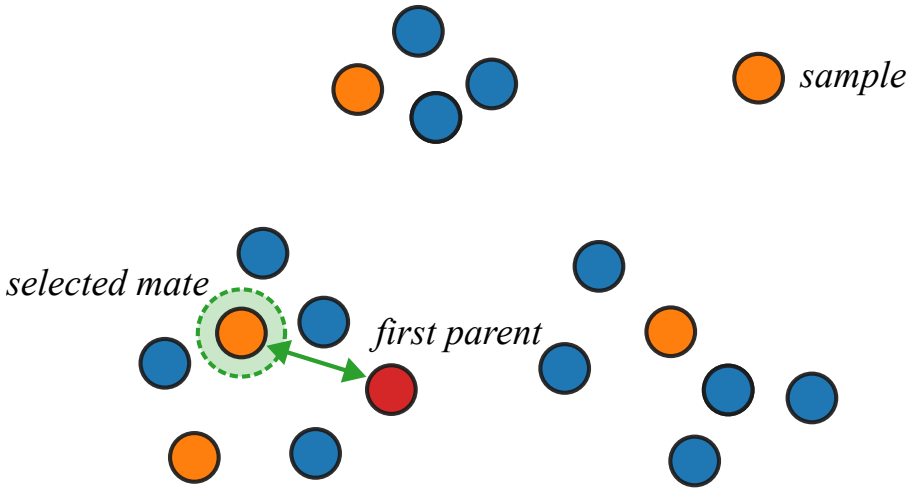


Figure 13: Approximate Nearest Neighbour Mate Selection. A first parent is selected (red). Then, a sample of the potential second parents is taken (orange). For each parent in the sample, the distance to first parent is calculated. The second parent with the closest distance (green outline) is selected to form a pair with the first parent. Note that this is (intentionally) not necessarily the true closest parent.

```
# Genome Distance Based Mate Selection through Approximate Nearest Neighbour

def select_reproduction_pairs(genome_population: list[Genome], n_reproduce: int
    , n_parent_pairs: int):
    # select the top ranked genomes for the set of reproducers
    parent_population = genome_population[0:n_reproduce]
    first_parents = select_first_parents(parent_population, n_parent_pairs)
    second_parents = [select_mate(p1, parent_population) for p1 in
        first_parents]
    return zip(first_parents, second_parents)

def select_first_parents(...):
    # Create a set of first parents with fitness weighted random sampling
    return first_parents

def select_mate(p1, parent_population):
    # Filter self and select a random sample
    eligible_parents = select_eligible_parents(p1, parent_population)
    p2 = min(eligible_parents, key=lambda p2: distance(p2, p1))
    return p2
```

Code 2: Genome Distance Based Mate Selection

4.3.4 Species Detection

Because speciation occurs bottom-up in a single population, methods to detect species are required. For the Toxin model, species are explicit due to their explicit and deterministic functions. Species labels can be derived directly by taking the binary agent chromosome and mapping it to a base 10 integer. Species labels for the optimization model should represent the different variable groupings,

so the species labels for those agents are obtained by converting the agent chromosome to a binary vector where each off bit is a knocked out gene, and a non-knocked out gene is an on bit, and mapping the resulting binary vector to a base 10 integer. In order to group the chromosomes together in reasonable groups that are similar, we can utilise the format of the chromosomes as vectors to apply a vector quantization technique. We can use unsupervised cluster detection algorithms. These cluster algorithms detect groups of similar datapoints. The analogy is that if we translate human DNA and the DNA of bananas to vectors, a cluster algorithm would be able to assign a group A and group B that correspond to the human DNA group and the banana DNA group.
In nature, the taxonomy of animals is also not always as clear-cut. During the process of speciation it is not always clear when the one species has become two, and the decision boundary is a human construct. There is within species variation too, and an individual with a particularly rare set of genes would not necessarily be considered a different species. This is a difficulty that we have to deal with, but it also affords us some freedom in ambiguous cases. It would be better suited to think of the gene pool with species as a collection of individuals that can loosely be grouped into sets of *quasispecies*. Over time, through positive feedback loops towards speciation, eventually species will clearly be distinct from each other.
In our gene pool, the differences wont be as clear, especially as the intent of the simulation is some form of speciation, meaning that there will be many in-between species forms and outliers.

There are many considerations when deciding which clustering algorithm to use, such as the speed, scalability, ability to detect non-spherical clusters, ability to deal with outliers, unbalanced cluster sizes. Careful consideration and testing has shown the DBSCAN algorithm to perform best in detecting clusters, as well as being scalable for a large number of datapoints and medium amount of clusters, as well as its ability to deal with non-flat geometry, uneven cluster sizes and ability to detect outliers.
DBSCAN (Density-Based Spatial Clustering of Applications with Noise)[85, 86] is a density-based cluster algorithm. It works by defining an epsilon value (ε) and a minimum number of points. Then it groups all datapoints together based on distance to other points. See fig. 14 for a visualization.

Two important parameters for DBSCAN are the ε value [133], which determines the maximum distance between two points for them to be considered in the same neighbourhood, and the minpts, the minimum number of datapoints a set should have for it to be considered a cluster. The first is a value that would need careful tuning for our case: set it too low and we will get many more species than desired. Set it too high, and we will not have enough definition of the species, and different clusters will be considered the same species. The second parameter, minimum of points, we will set to 1. In our model there are no outliers that are not also a species. A single outlier that does not belong to any other cluster should be considered its own species. The alternative is that all outliers belong to their own species group of outliers, which makes no sense.

See code 3 for a pseudocode implementation of the three species detection algorithms.

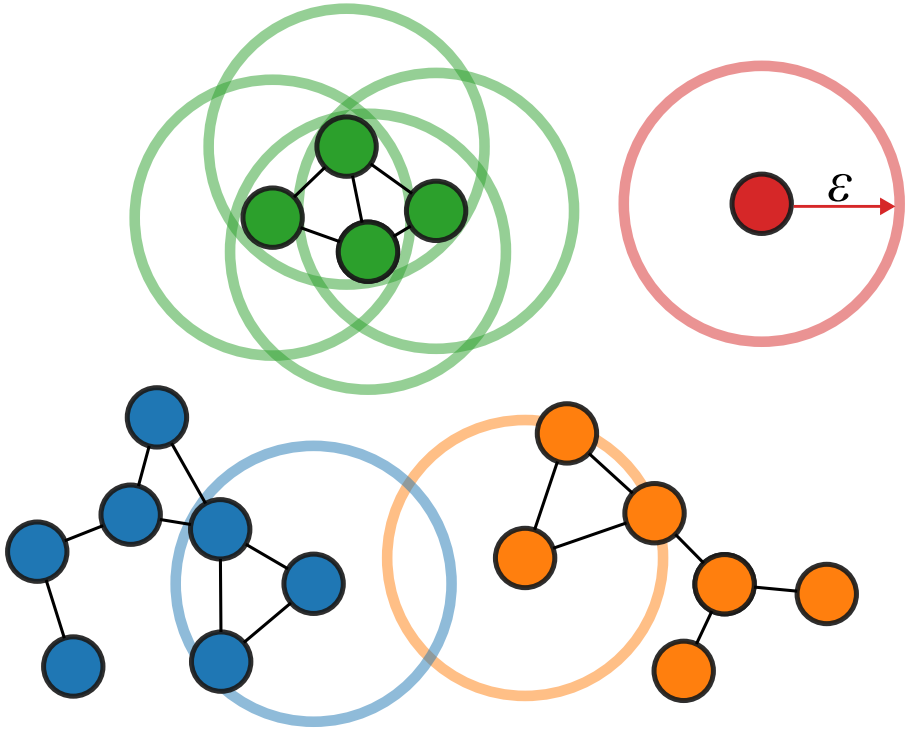


Figure 14: Cluster detection using DBSCAN. Clusters are build by connecting all datapoints that are within an ε radius of each other. All (indirectly) connected points are then assigned the same cluster label. The colours show the cluster labels. All green neighbourhood circles are shown to demonstrate when connections between points are made. The blue and orange neighbourhood circles show that overlap of circles do not warrant a connection. As the minimum number of points for a cluster is set to 1, the single datapoint is also assigned a cluster label.

```
# (Quasi)species detection for the chromosome population

def get_quasispecies(chromosome_population, species_type):
    if species_type == "binary" or species_type == "binary_knockout":
        quasispecies_labels = get_binary_quasispecies(chromosome_population,
            species_type)
    if species_type == "spatial_cluster":
        quasispecies_labels = get_spatial_cluster_quasispecies(
            chromosome_population)
    return quasispecies_labels

def get_binary_quasispecies(chromosome_population, species_type):
    quasispecies_labels = []
    for chromosome in chromosome population:
        if species_type = "binary":
            binary_chromosomes = [1 if gene > 0 else 0 for gene in chromosome]
        elif species_type = "binary_knockout":
            binary_chromosomes = [0 if gene = "knocked_out" else 1 for gene in
                chromosome]
        chromosome_label = binary_to_integer(binary_chromosome, base = 10)
        quasispecies_labels.append(chromosome_label)
    return quasispecies_labels

def gen_spatial_cluster_quasispecies(chromosome_population):
    # DBSCAN implementation of the scikit-learn (sklearn) library utilized.
    quasispecies_labels = sklearn.cluster.DBSCAN(epsilon, min_samples).
        fit_predict(chromosome_population)
    return quasispecies_labels
```

Code 3: Species Detection

4.3.5 Species Tracking / Label Homologation

Detecting species in the gene pool for a single generation is one thing, but it becomes more difficult when we want to track these species though time. Applying the same cluster algorithm to a new generation of chromosomes does not guarantee that the labels assigned are consistent with those of the previous generation. The label of a cluster can be arbitrarily reassigned from one generation to the next. We have to remember we're dealing with quasispecies, meaning that the individuals can change in steps from one generation to the next. Species assignment is context-dependent, and if many individuals change, the shapes of clusters will also change. To introduce some consistency, we apply '$g_{t-1} + g_t$' species detection; when performing the vector quantization both the current and previous generations' full population are included (see fig. 15).

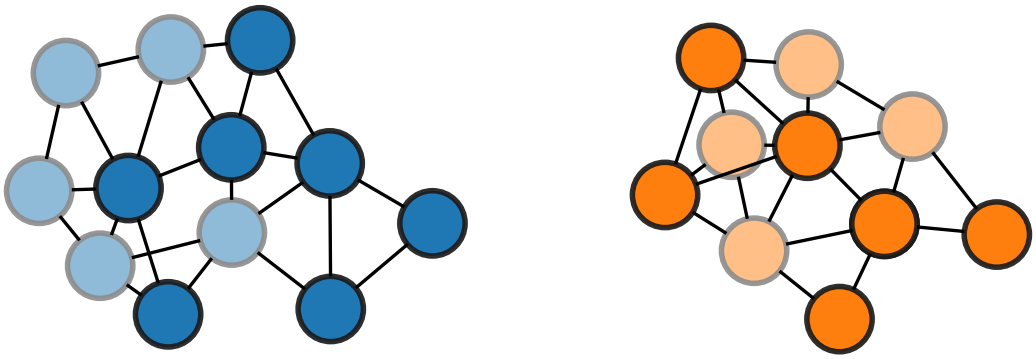


Figure 15: $g_{t-1} + g_t$ cluster detection: cluster detection is performed with both the current generation (opaque) and the previous generation (faded).

Additionally, the cluster prototypes are tracked for each generation. The cluster prototype type used are cluster centroids - the average position of all included datapoints. This is sub-optimal, as the DBSCAN algorithm is density-based and not centroid-based, however using density-based prototypes brings a lot of extra computational cost [134], and centroid prototypes have shown good performance during testing.

These prototypes are used to have label consistency. Each generation the cluster algorithm detects the current clusters. Next the cluster prototypes are calculate for each cluster (fig. 16). Then each prototype is assigned the label of the closest prototype from the previous n generations (fig. 17), if the distance is below a threshold (fig. 18a). A small n is desirable, setting a threshold ensures new species are allowed to emerge through genetic drift. If multiple prototypes in the newest generation are closest to the same older prototype (fig. 18b), only one assume the label, and the others are assigned a new label. Which prototype is assigned the old label is determined by checking which prototype is associated with the largest cluster (largest is assigned the label), and using the smallest distance to the cluster as a tie-breaker.

There will still be cases where one individual will be assigned species label A in one generation and species label B in the other, but using these species tracking techniques enables sufficient reliability over time. There is still some arbitrariness and stochasticity in this method, and further improvements to this method will most likely yield improved performance on the whole. Cluster tracking over time is an area of ongoing research that has picked up some interest recently with analyzing data patterns over time in streaming data such as social media [123, 124, 125].

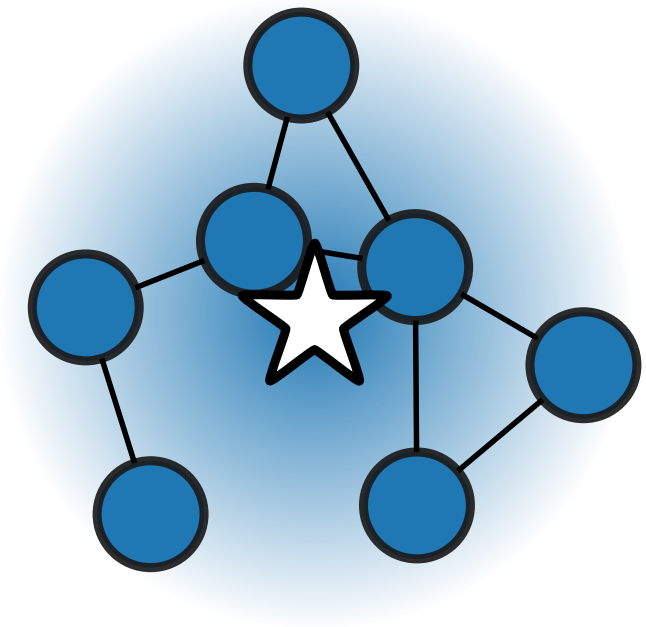
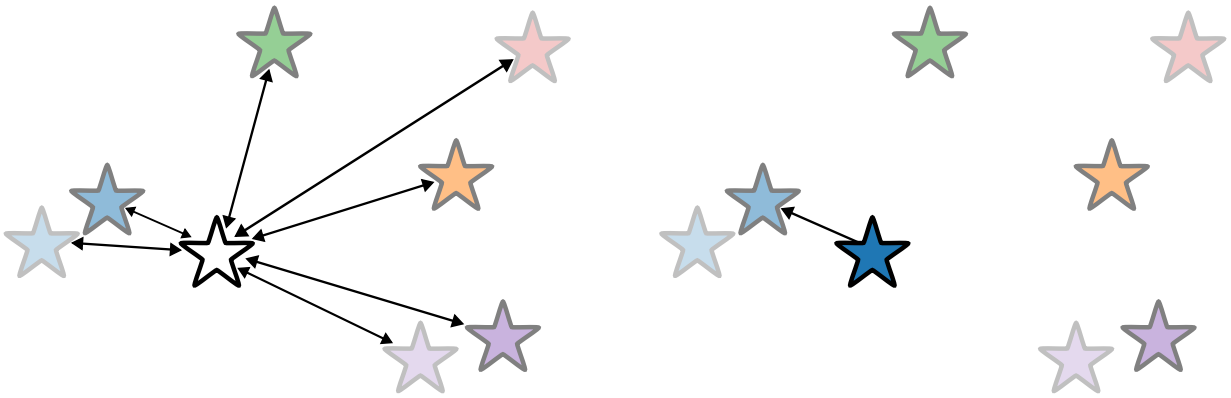


Figure 16: For each cluster, the prototype (in this case, centroid) is calculated. The cluster centroid is at the average position of all cluster constituent datapoints.



(a) The distance to previous n generations' known cluster centroids is calculated.

(b) The new cluster is assigned the label of the closest previously known cluster centroid.

Figure 17: To promote species label consistency, newly detected clusters are compared using cluster prototypes to previously known clusters. Labels are propagated through generations.

(a) A threshold value theta (θ) is set so any cluster that deviates significantly from known clusters is assigned a new label.

(b) If two new cluster prototypes are assigned the same closest known cluster, the cluster with the most constituents is assigned the label, and the other a new label. If both clusters are equally large, the closest prototype is assigned the label.

Figure 18: New species labels are created if there is no known previous cluster prototype within a threshold distance, or if there is a conflict over which new cluster should be assigned the same previously known cluster label.

```
def homologate_species_type_labels(clusters):
    # calculate prototype for clusters
    prototypes = [calculate_prototype(cluster) for cluster in clusters]

    # find closest known prototype of the last n generations, assign its label
       if it is within threshold distance, a new label otherwise.
    for p in prototypes:
        closest_known_prototype = find_closest_prototype(p, known_prototypes[-n
           :])
        if distance(closest_known_prototype, p) <= threshold:
            p.species_label = closest_known_prototype.species_label
        else :
            p.species_label = name_generator.next()

    # if the same label is assigned to multiple prototypes, reasign the labels
       of prototypes that are not the largest, or closest when multiple are the
        largest.
    for conflicting_prototypes in groupby(prototypes, species_label):
        if len(conflicting_prototypes) > 1:
            p = max(conflicting_prototypes, key=lambda p: (p.cluster_size, -p.
               distance_to_closest_prototype))
            others = conflicting_prototypes - p
            for other_p in others:
                other_p.species_label = name_generator.next()

    known_prototypes[generation] = prototypes
```

Code 4: Species Tracking Algorithm. In the true implementation, the steps are performed to build a translation key of the detected temporary labels to the most likely species type label considering previous cluster-label pairings.

4.3.6 Agent Sampling

In a domain model, a certain number of agents are required. For example in the Predator-Prey model, there are three predators, meaning there are three agent chromosomes required. We will refer to these positions as *agent slots*. A composition is the combination of chromosomes filling the agent slots in a single run of the model. Because the size of the population of agent chromosomes is larger than the number of slots, a sampling method is required to sample the individuals from the population. The sampling method should attempt to satisfy three constraints:

- Each individual chromosome is evaluated at least once.
- Chromosomes are sampled a proportionally equal amount.
- Chromosomes are sampled with as many unique others so the evaluation is robust.

A *simple random* sampling with replacement after exhaustion strategy adheres to the first two constraints. However, this strategy completely disregards the species type of the individuals, making it difficult for coevolution to occur.

Agents need to be evaluated under the conditions that they are 'supposed' to function. In a Predator-Prey simulation with a scenario with forced subpopulations, all compositions are A,B,C, meaning there will always be an agent of type A, an agent of type B, and an agent of type C present in the simulation. An A evolves in an environment with always one type B and one type C, and so allowing it to exploit the niche not present in the simulation. The species 'learn' to cooperate with each other as they are always part of each others environment. For this, a *deterministic sampling strategy* can be utilized.
When there are no forced subpopulations, we can not exploit this knowledge, as is the nature of having no fixed subpopulations. If we were to create groups of 3 by sampling randomly or uniformly from the population, the compositions would generally reflect the larger population distribution, but any species would be matched with any other species. Any fitness evaluation then would include a large number of combinations that it did not 'evolve' for, and subsequently they are not allowed to exploit a niche, but the fitness landscape would be such that the agent that can cooperate the best with any composition of other types would be the fittest. This does not promote niching towards speciation, but evolution towards generalists. To address this problem, the novel method of evolutionary compositions (section 4.3.7) is designed.

Deterministic sampling strategy If the population is segmented into equal sized forced sub-populations where the number of sub-populations corresponds to the number of slots, a *deterministic* sampling strategy can ensure all three constraints. The deterministic sampling algorithm iterates through the sets of chromosomes of distinct types and generates a specified number of combinations. It employs an index-based approach where from each subpopulation it selects the individual with the index corresponding to a_i in eq. (40).

$$a_i = c_i + (t_i \cdot l_i) \mod |S| \tag{40}$$

Here,

- a_i is the index of the agent in the population.
- c_i is the index of the current composition.

- t_i is the index of the type subpopulation.
- l_i is an iterator counting the number of loops done over the population.
- $|S|$ is the size of the subpopulations.

Specifically, this means the first loop produces $|S|$ compositions where the elements with the same index from the populations are sampled together. On the second loop, the indices of the second and third subpopulations are offset by $+1$ and $+2$ respectively, on the third loop the offsets are $+2$ and $+4$. This ensures chromosomes will be sampled at least once, equally as much, in unique combinations with as many other chromosomes. See code 5 for a *pyseudo*code example.

```
# Assumption: there are a fixed number of species, equal to the size of the
   composition, all with an equal number of members.

def generate_deterministic_chromosome_combinations(chromosomes, n_combinations)
   :

    chromosomes_per_type = index_chromosomes_per_type(chromosomes)
    n_chromosomes_per_type = length(chromosomes_per_type[type])
    chromosome_combinations = []
    loops = 0

    while True:
        for chromosome_index in n_chromosomes_per_type:
            chromosome_combination = []

            for type_index, chromosome_type in enumerate(chromosomes_per_type):
                selected_chromosome_index = (chromosome_index + (loops *
                   type_index)) % n_chromosomes_per_type
                chromosome_combination.append(chromosomes_per_type[
                   chromosome_type][selected_chromosome_index])

            chromosome_combinations.append(chromosome_combination)

        if len(chromosome_combinations) == n_combinations:
            return chromosome combinations

        loops += 1
```

Code 5: Deterministic Sampling Strategy

4.3.7 Evolutionary Compositions

To allow emergent species to coevolve, they need to be evaluated in composition with each other. If a fitness advantage of species A by exploiting a niche is dependent on the presence of another species B, evaluating species A without species B present gives poor results, giving species A a selection disadvantage. Beyond needing a single other species to be present, the evolutionary landscape is determined by the complete environment, and relationships can be indirect and complex. It could be

possible that the fitness advantage of A enjoyed with a B present is only possible if species B expresses a behaviour made indirectly possible by a species C, or if there are more than one of species A. Therefore, in order to properly evaluate species, they need to be evaluated in compositions that allow them to exploit their niche. In other words, there needs to be some stability in the evolutionary landscape for agents to move into their niche. This problem is non-trivial, because both the number of individuals of a certain type is variable, the number of types is variable and not restricted to the number of slots, and most importantly, the types themselves are emerging and disappearing.

Creating a configuration of variables in order to optimize an objective value is an optimization problem, and so the solution we propose is to determine the compositions using an evolutionary algorithm.

A composition chromosome is a combination of agent chromosome types, that can be evaluated by using the collective fitness of the agents in the domain model. Note that the chromosome is explicitly a combination and not a permutation. The chromosome is a vector where each value, representing a gene, is a nominal value signifying a species type, e.g., $C = [A, A, B]$. For the Predator-Prey model, the gene is the detected species label (section 4.3.4). For the Toxin model, the species label is the base 10 integer value of the binary agent chromosome. For the optimization model, the species label is the integer value of the agent chromosome mapped to binary, where a non-knocked out gene is 1, and a knocked out gene is 0 (meaning variable groupings determine the species). The phenotype of this chromosome is the expression of a composition of sampled agents with the corresponding type in the domain model. Figure 19 visualizes how an evolutionary compositions chromosome is used as sampling strategy, selecting agent individuals of types corresponding to the genes on the composition chromosome to perform in the domain model. Using a genetic algorithm to determine the compositions has the advantage of dynamically adjusting the compositions when new dynamics emerge. It balances exploitation of known well performing global and local (subcomponent) compositions and exploration by recombination and variation. By providing some stability of composition it facilitates coevolutionary growth. By recombination and variation it explores potentially beneficial (sub)compositions and allows a flexibility to move with a dynamically changing system.

There are also some caveats that we need to account for. The composition chromosome population and the agent chromosome population are decoupled. A next generation of the composition chromosomes are an 'improved' generation of compositions, optimized for the *previous* generation of agent chromosomes. The compositions are trying to hit a moving target.

The types have different rates of change between populations. In the agent population the proportion of chromosomes with a specific type can change strongly between two generations. The number of genes signifying that species type maximal change between two generations of composition chromosomes is lower. This recalcitrance in change is something that can be used as an advantage. More on that advantage in section 4.3.8.

Composition-Population alignment Because the populations are decoupled, it is not possible for them to be perfectly adjusted to each other at all times. It is possible that a type featured in the compositions has gone extinct in the agent population, and if a new type emerges in the population it needs to be introduced in the compositions, and there need to be sufficient slots for all chromosomes to feature in at least one composition. This can be solved in two ways, and which way constitutes a big design decision. The compositions can be aligned with the agent population, or the compositions can

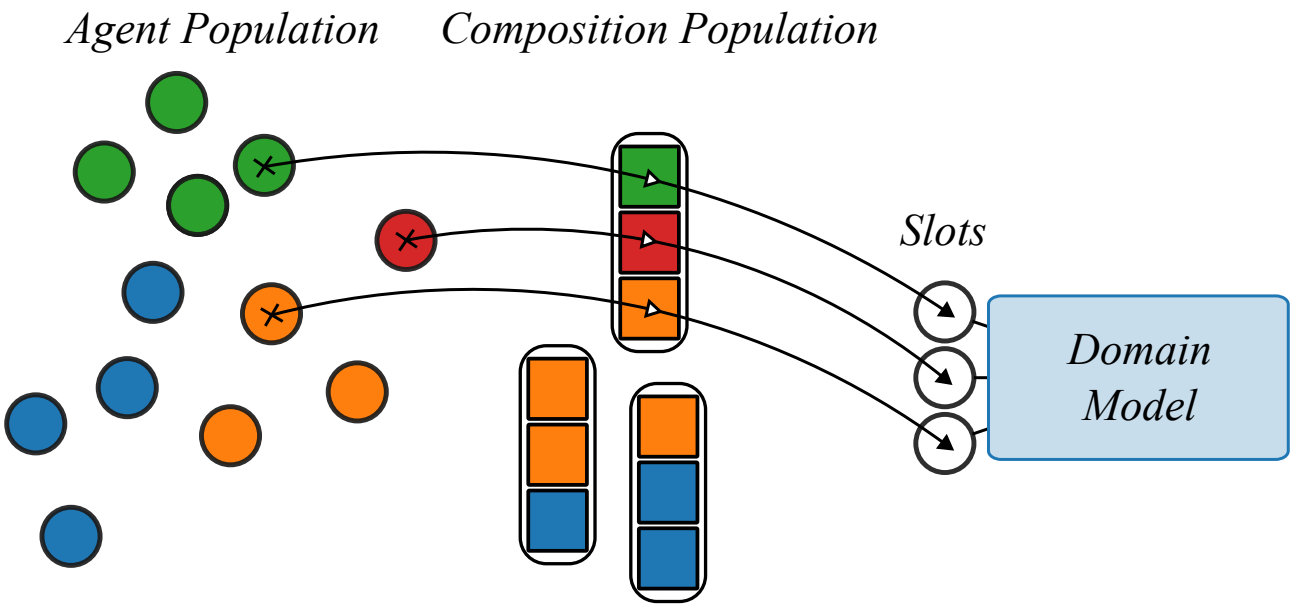


Figure 19: Evolutionary agent sampling creates compositions of individuals according to their type, directed by composition chromosomes. In this schematic overview, the domain model requires three individuals to function (three slots), as is the case in the Predator-Prey model. Here we see three composition chromosomes, where the genes signify the agent type. The green, red and orange composition samples one agent of each of those types, and delivers them to the domain model.

be taken as leading. Favouring the agent population entails adjusting the composition chromosomes to adhere to constraints set by the agent population. Favouring the compositions will result in some agents from the population not being selected for any composition, which effectively means they are culled without evaluating their fitness.

The composition chromosomes are aligned with the agent chromosome population in two ways. The evolutionary step of the composition chromosomes occurs after the evolutionary step of the agent chromosomes, so the new species types and their corresponding counts are known. The first is that during evolutionary step of the composition chromosomes when a mutation occurs, a mutation is restricted to the set of species types occurring in the new agent chromosome population. Genes on a composition chromosome signifying a species type no longer present in the agent chromosome population are also mutated. This ensures that all compositions only feature species types that are present in the agent chromosome gene pool. The second alignment strategy ensures that there is at least one available slot for each agent chromosome. After the evolutionary steps of both chromosome populations, the differences between each species type occurrence are calculated. A negative delta means that the type occurs more in the chromosome population than in the compositions, and so the negative delta correspond to the number of agent chromosomes for which there is no slot available. The composition chromosomes are iterated over in random order and the most over-represented genes are replaced by underrepresented genes until there are no negative deltas left. See code 6

To favour the compositions over the agent population, the above alignment strategies are simply not applied. The new generation of agents is generated with the remaining agents that did feature in compositions. In this case, the values for *cull fraction* and *reproduce fraction* are applied to the desired population size, not the actual number of agent chromosomes.

By employing evolutionary compositions with an agent population and a population for compositions, both concurrent learning and a hybrid heterogeneous team learning is employed (section 3.2). The

next section describes how the credit assignment problem is dealt with.

```
def align_compositions_with_chromosome_population_types(compositions,
    chromosomes):
    type_deltas = get_chromosome_composition_type_deltas(compositions,
        chromosomes)

    # Get transformations
    transformations = []
    while any(delta < 0 for delta in type_deltas.values()):
        from_type = max(type_deltas, key=type_deltas.get)
        to_type = min(type_deltas, key=type_deltas.get)
        type_deltas[from_type] -= 1
        type_deltas[to_type] += 1
        transformations.append((from_type, to_type))

    # Apply transformations
    for from_type, to_type in transformations:
        # Loop over the compositions randomly so we don't introduce a bias with
            changing earlier compositions
        shuffle(compositions)
        for composition in compositions:
            if from_type in composition:
                composition.remove(from_type)
                composition.append(to_type)
                break
    return compositions

def get_chromosome_composition_type_deltas(compositions, chromosomes):
    # Check the differences between type occurance in the chromosome population
        and the compositions.

    # Extract chromosome types from chromosomes and compositions
    chromosome_types = [chromosome.type for chromosome in chromosomes]
    composition_types = [type for composition in compositions for type in
        composition]

    # Count occurrences of each type in chromosomes and compositions
    chromosome_counts = Counter(chromosome_types)
    composition_counts = Counter(composition_types)

    # Calculate and return the deltas between compositions and chromosome
        counts
    type_deltas = {type: composition_counts[type] - chromosome_counts[type] for
        type in chromosome_counts}
    return type_deltas
```

Code 6: Agent population - Compositions Alignment

4.3.8 Species Representation Fitness Scaling

While there is now evolution present on both the individual and composition level, the individual population evolutionary process is still completely decoupled from the compositional evolution. Say there is an individual species type that performs a function at personal cost, but yields a net gain for the population. This species has an evolutionary disadvantage to its competitors in the agent population. Compositions are evolved with team performance, and so will feature species that contribute to the collective. This means that a species with self-sacrificing altruistic behaviours feature reliably in compositions, and decline in the agent population. Those species are represented proportionally more in the compositions than in the individual population. This signature of altruistic species can be utilized. By introducing a fitness advantage for species that are represented more in the compositions than in the population, species that provide a public goods at personal cost are compensated for this cost. In this way, a personal cost is allowed so far as it benefits the collective. To this end, we introduce the *species representation fitness scaling* mechanism. The personal fitness of each individual is scaled with a scalar that is equal to the representation rate of its species in the compositions. This introduces an evolutionary pressure on the individual population to shape according to the distribution of the compositions. Compositions are evolving for teamwork, individuals evolve competitively.

Given,

- A The set of agents $\{a_1, a_2, ..., a_n\}$.
- a_i Agent a_i with a species type $\text{type}(a_i) \in S$ and a fitness $\text{fitness}(a_i) \in \mathbb{R}$.
- C The set of compositions $\{c_1, c_2, ..., c_m\}$.
- c_j Composition c_j is a multi-set of species types, e.g., $[s_1, s_2, s_2]$.
- S The set of unique species types $\{s_1, s_2, ..., s_k\}$.
- s_k Species type identifier s_k.

First a normalization factor α is calculated with eq. (41) to account for differences in agent population and total slots in the composition population.

$$\alpha = \frac{|A|}{\sum_{j=1}^{m} |c_j|} \tag{41}$$

Then, for each species type $s \in S$ the fitness scaling factor β_s can be calculated using eq. (44).

$$\text{count}_C(s) = \sum_{c_j \in C} |[s : s \in c_j]| \tag{42}$$

$$\text{count}_A(s) = |\{a_i : \text{type}(a_i) = s, a_i \in A\}| \tag{43}$$

$$\beta_s = \alpha \times \frac{\text{count}_C(s)}{\text{count}_A(s)} \tag{44}$$

Finally, the fitness of each agent a is scaled according to its species scaling factor with eq. (45).

$$\text{fitness}_{\text{scaled}}(a_i) = \text{fitness}_{\text{base}}(a_i) \times \beta_{\text{type}(a_i)} \tag{45}$$

It is possible that there are multiple niches that can coevolve. It is very possible and probably very likely that not all niches on the sequence space evaluate to the same range of fitness. Other than in nature, the artificial selection performs selection on one fitness range, agnostic of species or niche. Any species that has a lower fitness will be out-competed by another species with a higher fitness. Even if both species are in an obligate mutualistic relationship, the species with the higher fitness will crowd out the lower fitness species, causing the extinction of the lower fitness species, which means that the higher fitness species crashes. This is the difference in fitness that should be compensated for with the species representation fitness scaling.

4.3.9 Knockout mutation

For the Function Optimization domain model, we want agent chromosomes to be able to represent any possible subcomponent / variable grouping. Species should be allowed to evolve to include more or fewer variables in its grouping. A full solution consists of a combination of agent chromosomes, and the solution vector is constructed by taking the sum of all chromosome vectors. If the value of a gene is 0, that gene has no effect on the solution vector. This means that by setting a gene value to 0 is equivalent to removing that component variable from the subcomponent. To facilitate dynamic variable grouping and subcomponent size, the gene knockout mutation mechanic is introduced in the evolutionary step of the architecture. Each gene has a probability *knockout mutation probability* to be knocked out, meaning its value is set to 0 and no further mutations are allowed except the un-knockout mutation, which sets the value of the gene to the value of any random other gene on the chromosome and re-allows regular mutations to occur. During the crossover step, knocked out genes are transferred to offspring in the same way as non-knocked out genes.

4.3.10 Chromosome resizing

When there is no predetermined number of slots in a domain model as in the Function Optimization model, the there is no size restriction on the composition chromosome. Therefore the composition chromosome is allowed to resize through a resizing mutation. In essence this mutation is similar to the knockout mutation, but since a composition is a combination and not a permutation a resizing mutation can be simpler, by either removing or adding a random gene with equal probability. Removing a gene simply removes that gene from the chromosome, adding a gene inserts a gene at a random position with either the value of a random other gene on the chromosome, or a value from the list of allowed mutations.

4.4 Parameters and Configurations

An overview of the parameters and model configurations is provided in tables in the appendix B, along with their effect and default values.

5 Experimental Setup

5.1 Research Questions & Hypotheses

To reiterate the primary research question first stated section 1.2 that this research explores:

Research Question Can evolutionary pressures within a single gene pool lead to the emergence of mutualistic species that collaboratively solve complex problems in dynamically composed teams?

To address this question, the following hypotheses and sub-hypotheses are investigated:

Hypothesis 1 Multiple species can coexist stably in a single chromosome population, maintaining functional heterogeneity within a single genome population.

1.1 Genome distance-based mating serves as a dynamic assortative mating mechanism for speciation.

Hypothesis 2 A composition of individuals from multiple species can collectively form a complete solution to a problem, with each species contributing a distinct subcomponent.

2.1 These species form an obligate mutualistic relationship and are interdependent.

Hypothesis 3 The emerging species converge to a near-optimal decomposition of the problem.

Hypothesis 4 A hybrid approach combining individual and team level selection promotes the survival of self-sacrificing altruistic behaviour in a competitive environment.

Hypothesis 1 *"Multiple species can coexist."*
First, it should be proven a stable coexistence of multiple species with distinct behavioural functionality can emerge and be maintained within a single genome population. The main mechanic combating local cluster diffusion and single group convergence is the dynamic fuzzy compartmentalization of genome distance-based mate selection. The validity of this hypothesis can be shown by showing speciation occurs, and comparing speciation in model runs with and without the mechanic active.

Hypothesis 2 *"The species can cooperate."*
Once it is established that multiple distinct species can evolve from a single population, the next step is to show that the species are symbiotic, and in composition form a solution to the fitness criterion. To prove that species have coevolved cooperatively, it should be shown that they form a solution when in a composition with each other, but not independent. This can be done by measuring a metric of problem solved-ness for the different domain models, and showing that when a species is forcibly removed the remaining species fail to perform.

Hypothesis 3 *"The best teams are selected."*
Considering evolutionary pressures on the individual promote efficient individuals, and evolutionary pressures on compositions promote efficient compositions, it is worthwhile to explore if the full solution is optimal. When the sequence space is large, it should not be expected that an evolutionary algorithm always finds the global optimum. This hypotheses can be tested by comparing the emergent composition to a known optimal decomposition of the problem. Such an optimal decomposition is

most explicit for the Toxin model, where the theoretical optimal is any composition where the total system fitness equals the maximum social welfare Pareto-optimum. If there are redundant functions, the solution is not optimal. In the Function Optimization model, linked decision variables should group in the same species.

Hypothesis 4 *"Team-players are recognized."*
A competitive environment where each individual seeks to maximize its fitness eliminates strategies that produce additional public good at personal cost. Composition evaluation shows the total public good produced by the combination of individuals. Including a heuristic measuring the public good contribution at personal cost for individuals (section 4.3.8) and 'compensates' individuals proportionally for that cost, allows altruistic strategies to survive. This can be tested by seeing if self-sacrificing altruistic individuals are part of the converged solution composition.

5.2 Experiments

While variation exists for specific experiments, the main experimental framework for testing the hypotheses are comparing three distinct configurations on the domain models:

RC Random compositions: traditional EA configuration with random agent sampling and fitness proportional mate selection.

FS Forced subpopulations: traditional CCEA configuration with forced subpopulations and fitness proportional mate selection.

EC Evolutionary compositions: evolutionary composition agent sampling and genome distance based mate selection.

Comparing these three configurations gives the clearest view of the properties of the proposed methods.

Table 4 shows which domain models are used in experiments to test the hypotheses. Hypothesis 1 benefits from the open-ended nature of the solution space of the solution space of the Predator-Prey model, in which a collapse to a single species is most likely. The results of experiments for other hypotheses already show the ability of the model to maintain coexisting species in the Toxin and Function Optimization domain models. The effect of genome distance based mating is observed by comparing chromosome groupings in a 2D mapping of the many dimensional chromosomes between the three configurations of compositions. This compares no genome distance based mating in a single population with no compartmentalization (RC) and total compartmentalization of three populations (FS) with genome distance based mating in one population (EC).
That the emergent species can cooperate should be a universal outcome of the model, and so hypothesis 2 is tested for all three domain models by comparing the quality of produced solutions across the three configurations. The performance of all three domain models can be measured, the Predator-Prey model by its benchmark, the Toxin model by the remaining toxin in the environment, and Function Optimization model's ability to find the solution vector that minimizes the objective value. Whether the emerged mutualistic relationship is facultative or obligate is tested by artificially removing all individuals of one of the dominant species (with the largest population size) at a set timestep, and observe the change in performance of the remaining compositions.

Hypothesis		1	2	3	4
Domain Model	Predator-Prey	X	X	X	X
	Toxin		X	X	X
	Function Optimization		X	X	

Table 4: Domain Models used to test each hypothesis.

To test hypothesis 3 we look at the optimality and decomposition of the solution, and compare it to the theoretical optimum. In the Toxin model this means creating an evolutionary pressure for the grouping of functions by adjusting the function cost multiplier parameter M. For the Function Optimization model we look at the subcomponent size and number of subcomponents, compared to the theoretical optimal decomposition. An optimal decomposition in the Predator-Prey model constitutes a combination of specific behaviours, which are difficult to analyze, but by looking at the number of emergent species and comparing the achieved benchmarks to a forced subpopulations approach some inference can be made about the optimality of the emergent decomposition.
Hypothesis 4 is tested on the Toxin model, where functions of agents and their contribution to personal and collective gain are explicit. A baseline of the dynamics is set with random compositions. The effect of the fitness scaling mechanism is observed by comparing equal runs using evolutionary compositions with and without the mechanism active, specifically if the converged equilibrium is the Pareto-optimal social welfare. A comparison is made to a run without fitness scaling where the individual fitness is wholly determined by the collective fitness to show how fitness scaling is similar to but not exactly the same to optimizing for the collective welfare, and a Predator-Prey model run with these same parameters is run to determine how these dynamics hold up in a larger search space where species can go extinct more freely.

The default (hyper)parameters used throughout experimentation can be found in appendix B. The specific setup and complete results for each experiment, along with the source code of the implementation, can be found on the GitHub page found in appendix C, as well as a 'ReadMe' file detailing the technology stack and instructions on how to run an experiment.

5.3 Hyperparameter Justification

Experimental disclaimer: The model architecture is large, and there are many moving parts that all deserve their own analysis and experimentation. It is infeasible to test and minutely justify each parameter choice. The objective of the thesis is to show that it is possible for symbiotic species to emerge from a single population, the parameter choice serves to facilitate that the desired effect appears. In order to limit the scope of the research, some constituent parts will need to be justified theoretically rather than experimentally, and experimentally only in combination with everything else. There is an operating window, or range of parameters, in which the effect can appear. It is not the aim of this research to prove the viable range of these parameters. There will no doubt be more efficient parameters for domain specific applications.
For all parameters, a course and fine grained parameter sweep is performed. Results of these, along with many variations of the discussed experiment results, can be found in the code repository which can be found in appendix C.
For the Predator-Prey model, results of single runs are displayed to give better insight in what is hap-

pening in the model. Emerged species (labels) are unique for their run, so averages of species across multiple runs is an insufficient metric. For the Toxin and Function Optimization model species labels are consistent, and so average results of 10 runs are shown. These values and the other parameters in table 6 are chosen to give sufficient 'room' for the model to work, while keeping computational space and time complexity at a manageable level. When experiment parameters deviate from these values, that is explicitly mentioned.

6 Results

Due to the inherent stochasticity in evolutionary processes and the model's design, individual runs with identical parameters can yield a diverse array of outcomes. Nonetheless, the results presented here are representative of the general patterns and effects consistently observed across a substantial number of simulations. Results are often presented in terms of averages in order to show general converged trend lines, which is sufficient to show qualitative congruence with the theory. The focus of the results is on showing these qualitative dynamics, and therefore some liberties have been taken to improve visual clarity in lieu of exact correctness. Most notably the presented results are often smoothed running averages, the size of the spread of results is omitted since these are dependent on parameter specifics, and legends are limited to the top relevant species. To reiterate, more results for each experiment and many more can be found on the GitHub page linked in appendix C.

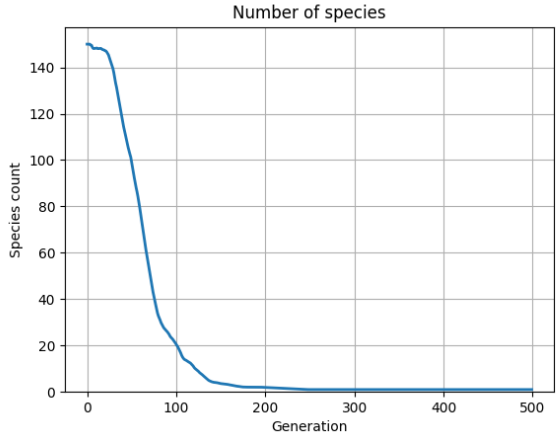
6.1 Hypothesis 1

Hypothesis 1 Multiple species can coexist stably in a single chromosome population, maintaining functional heterogeneity within a single genome population.

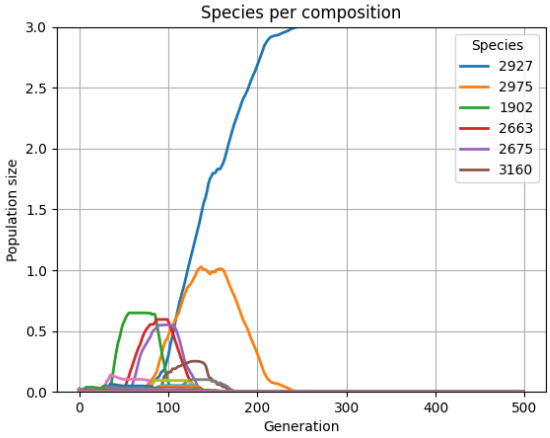
1.1 Genome distance-based mating serves as a dynamic assortative mating mechanism for speciation.

6.1.1 Predator-Prey

To show that multiple species can coexist stably in a single population we compare the evolution of species in the Predator-Prey domain model. Using the traditional evolutionary configuration where the compositions are sampled randomly from the population, and mate selection is done proportional to fitness (RC), we observe in fig. 20a that the number of species collapses to a single species. This results in fully homogeneous compositions. In contrast, when using evolutionary compositions and genome distance based mate selection (EC), fig. 21a shows that two species can survive in the same population. Figure 21b shows the average number of individuals per species in a composition. Since in the Predator-Prey model there are three foxes, a composition consists of three agents. The graph shows that compositions are formed of two species, on average equally many per species. Note the graphs display a smoothed moving average for readability, oscillations have higher amplitude and frequency than visible. The graph indicates that the composition oscillates between 'AAB' and 'ABB' configurations, where A and B signify the species. In this specific graph, A and B are species '340' and '371'. Species are named in order of first detection. The oscillatory behaviour and the fact that two species survive instead of three or one can point to that there is no need for a third species - two strategies in whichever composition are sufficient. It is also possible that this composition is a local optimum that the model can not get out of. In a minority of runs of the model, results not included here, three species emerge and form a stable composition.
For an insight in what species look like a lower (2) dimensional representation of the 22-dimensional chromosome vectors is shown in fig. 22 using t-SNE. A reference image fig. 22a shows final generation vectors for a forced subpopulations run, where the number of subpopulations is set initially and compositions are created with deterministic sampling. Clear clusters are visible. Figure 22c shows the spatial structure of the single remaining species for the last generation of the model run using traditional evolutionary parameters. Figure 22b shows the clusters of the remaining two species in the evolutionary compositions with genome distance based mating run. Here we see three clusters of vectors, identified as two distinct species. That two clusters in this lower dimensional representation



(a) The number of species in the agent population (single run). Decreasing to one species over generations.



(b) The average number of agents of a species per composition (100 compositions). The compositions converge to consist of only a single species.

Figure 20: The number of species in the agent population (a) the species composition composition (b) for the Predator-Prey model for RC configuration.

are identified as the same species could possibly be explained by the fact that it is a lower dimensional representation, and that in higher dimensions the clusters appear relatively more similar. Another explanation is that the ε value used for the DBSCAN cluster detection could be fine-tuned more. It also shows that the clustering effect of genome distance based mating is separate from species classification.
We see that when genome distance based mating is used, species are more consistent – chromosomes cluster and so species are less diffused than in fig. 22c. We can see this too with the species naming in the legend. Each detected species is given an integer value as name, starting at 0 and counting incrementally for each new species. The high value for species names (in fig. 22c) indicate that many species have been recognized.

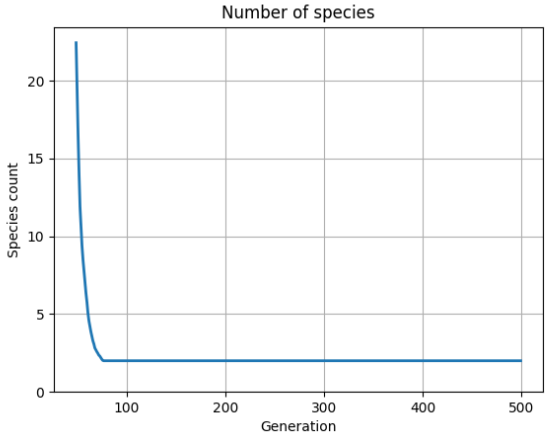
6.2 Hypothesis 2

Hypothesis 2 A composition of individuals from multiple species can collectively form a complete solution to a problem, with each species contributing a distinct subcomponent.

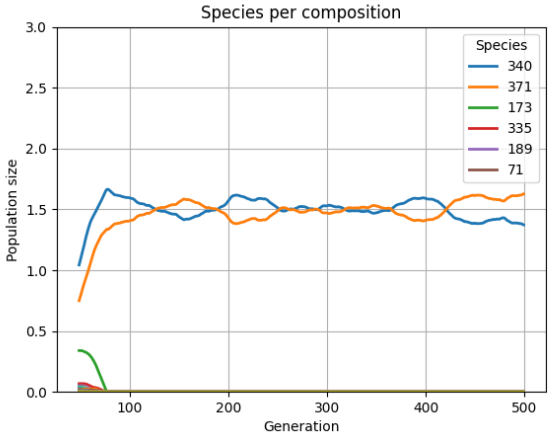
2.1 These species form an obligate mutualistic relationship and are interdependent.

6.2.1 Predator-Prey

The baseline benchmarks of the forced subpopulation configuration in fig. 23a show that when each agent type is evolved in isolation and the number of subpopulations is equal to the number of slots in the domain model, a fairly consistent maximum score can be attained. Three distinct and separate fox types coevolved to cooperate with the other two form an effective team for catching prey. The traditional random sampling and fitness proportional approach that collapses to a single species in fig. 23b shows that a fully homogeneous composition performs worse. Three agents of the same type can still catch the prey fairly consistently, but are not as effective as three distinct behaviours. Figure 23c shows that the two species evolved with evolutionary compositions and genome distance based mating



(a) Number of species in the agent population. Decreasing to two species over generations.



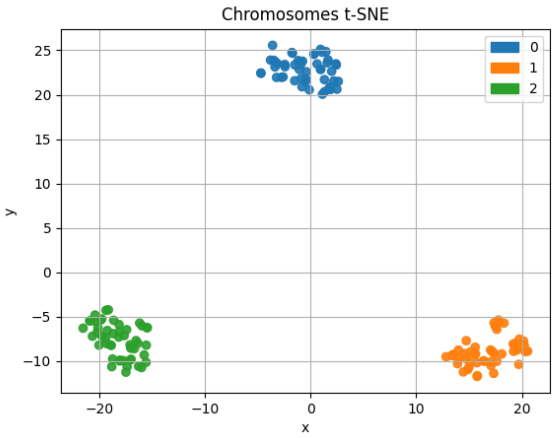
(b) The average number of agents of a species per composition. The compositions converge to consist of two species, oscillating between 'AAB' and 'ABB'.

Figure 21: The number of species in the agent population (a) the species composition composition (b) for the Predator-Prey model with evolutionary chromosome sampling and genome distance based mate selection (EC) configuration.

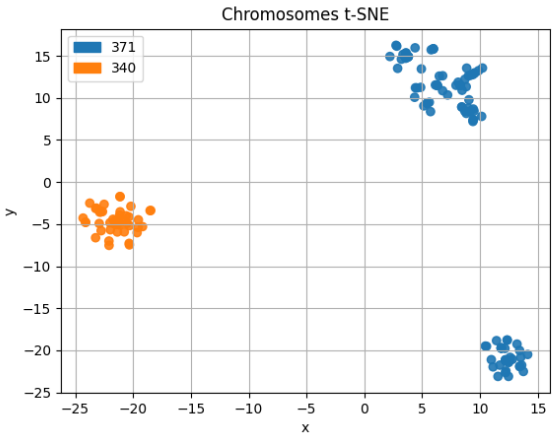
are effective at catching the prey, though slightly worse and less consistent than the forced subpopulations approach. It is distinctly better than the single genome population that converges to a single solution. Figure 23d shows what happens to the performance if all individuals of one species are (manually) removed from the simulation at generation 250. The two species have formed an obligate mutualistic relationship, and removing one from the environment causes the performance of the other to collapse, it is incapable of catching the prey in a homogeneous composition of only its own type. Over time, the species behaviour is adjusted to fit the new environment and performance improves. In fig. 24 the species compositions shows that a new species emerges that benefits from the vestigial behaviour of the lone surviving symbiont. When that new species proliferates, opportunity for a new niche better suited as symbiont for the new species allows a second new species to emerge. The two new species initiate a mutualistic relationship and coevolve. Note that the prototypical genotype of a species is allowed to shift over time, and that a new species emerging constitutes that a new genome is sufficiently far in genomic distance to existing prototypes. This can be due to sufficiently large mutations, or crossover between parents of different species producing offspring with sufficient genomic distance to either parents' species prototype.

6.2.2 Toxin

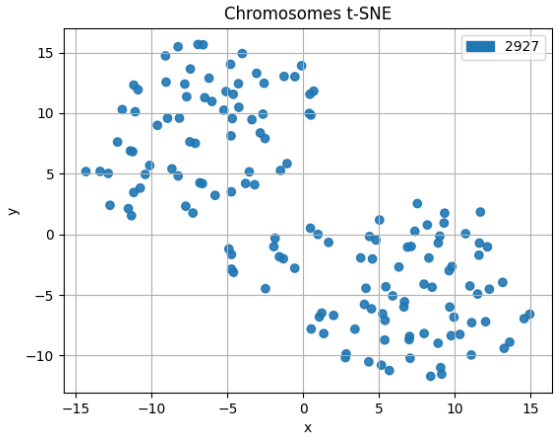
In fig. 25 a measure of 'problem-solvedness' of the Toxin model is displayed, namely the level of remaining toxin in the environment for each toxin. The base toxin level value is 6, with 5 toxins, and 30 agents per composition gives that every agent needs one active function on average for all toxin to be cleaned. The gene function cost is 3, meaning the break-even point for an agent to adopt the cleaning function for a specific toxin is if the toxin level of that toxin in the environment is 3. Note that agents are evolved with their personal fitness, and compositions with the collective fitness. When agents are evolved using the collective fitness (e.g., $cfw = 1$ with the toxin remainder, results in appendix C), all configurations manage to decrease toxin levels to 0. The traditional evolutionary parameters using random sampling and fitness proportional mate selection in fig. 25a and the forced subpopulation



(a) Species chromosome clusters for the Predator-Prey model run using forced subpopulations (FS).

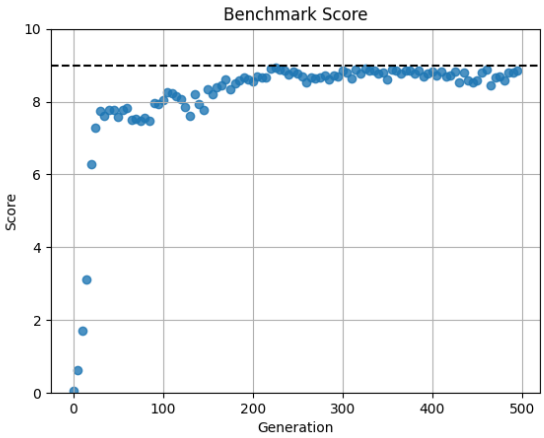


(b) Species chromosome clusters for the Predator-Prey model run using evolutionary compositions and genome distance based mate selection (EC).

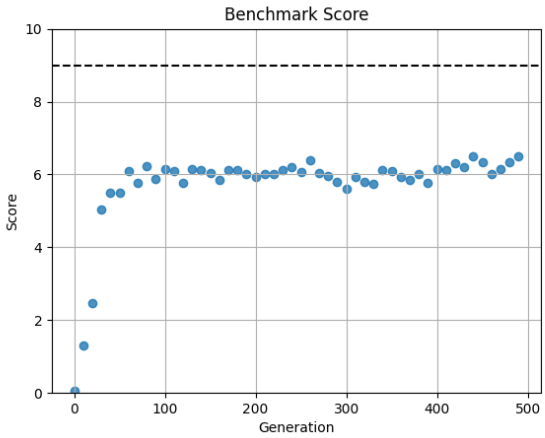


(c) Species chromosome clusters for the Predator-Prey model run with random sampling and fitness proportional mate selection (RC).

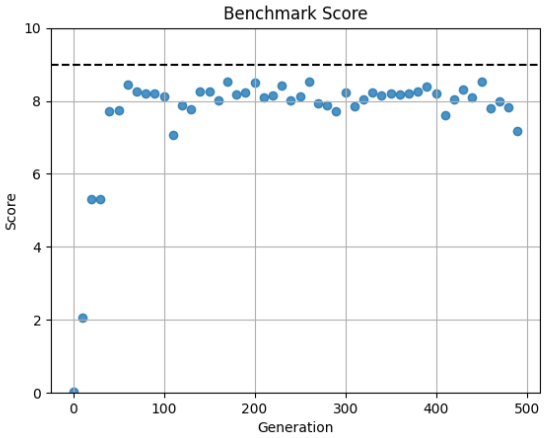
Figure 22: t-SNE plots of agent chromosomes of the final generation of a Predator-Prey model run. A lower dimension mapping of chromosome vectors. Colours indicate detected species.



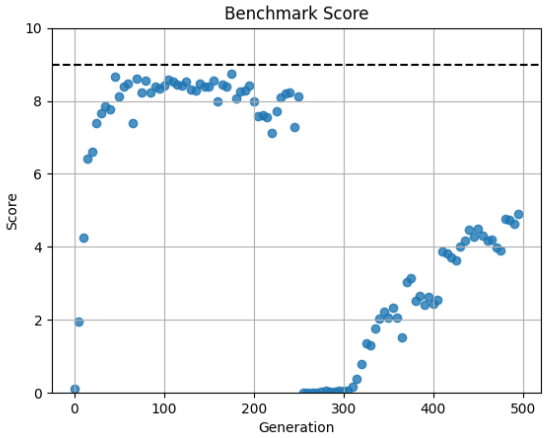
(a) Baseline benchmarks using FS configuration.



(b) Baseline benchmarks using RC. The population collapses to a single species, compositions are homogeneous.



(c) Benchmarks with EC.



(d) Benchmarks with EC. At generation 250 an extinction event is triggered, removing all chromosomes of one type.

Figure 23: Benchmark results of various Predator-Prey domain model configurations. The dotted line indicates the maximum score of 9/9.

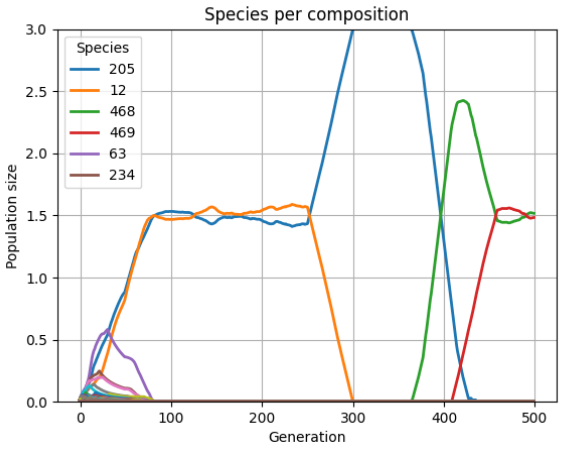
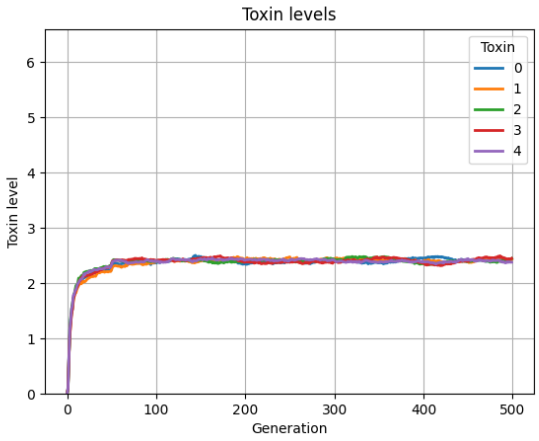
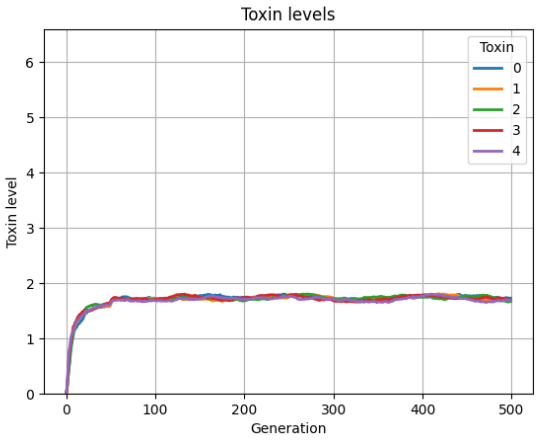


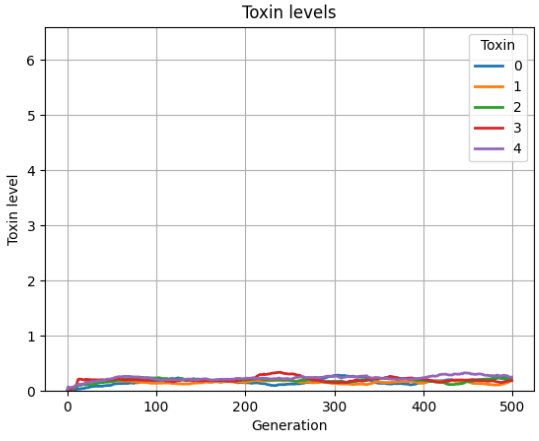
Figure 24: The average composition of species where a single species (the species with label 12) is eliminated at generation 250. The gradual decline seen is an artifact of the line smoothing using a moving average. Results from the same run as fig. 23d.



(a) RC, toxin levels



(b) FS, toxin levels



(c) EC, toxin levels

$$
\begin{aligned}
|T| &: 5 \\
|A| &: 30 \\
c_B &: 6 \\
R &: 1 \\
C_f &: 3 \\
M &: 1
\end{aligned}
$$

(d) parameters

Figure 25: Average toxin levels for various configurations in the toxin domain model with 5 toxins. Lower values mean more toxin is cleaned from the environment, lower is better.

configuration in fig. 25b are unable to decrease toxin levels to a minimum. Figure 25c shows that evolutionary compositions and genome distance based mating are more capable of decreasing toxin levels, when fitness of individuals is determined by their personal fitness.

6.2.3 Function Optimization

For the optimization model, the metric showing the level of problem-solvedness is the objective function value. The objective value is the output of the optimization function for the given input vector, which is composed of a composition of agent chromosomes and a standard offset applied, see section 4.2.3. Figure 26 shows the average objective value for the four optimization functions (eqs. (20) to (23)) for the three model configurations. The trend observable is that for all functions except SPS, for all model configurations the model is able to converge to the optimum in about the same number of generations. The FS configuration takes a long time to get to the true optimum for the Sphere and Griewangk function after quickly converging to a near optimum value. All of these graphs (except SPS) show a similar curve shape, starting with logarithmic decline of the value until it starts to level off (and in some cases plateaus), followed by a kink angled down, then turning in the optimal value or an asymptote approaching the optimal value. The model has difficulty solving the SPS function. EC decreases the value, and then improves very little. RC performs best and shows the same curve as for the other functions, but ends on a higher value than the optimum. FS seems to be unable to solve the problem. Problems with progressive optimization could be explained by the partially separable nature of the function. Another explanation for the difficulty in solving this function is that the optimum is in a small range ([-1, 1]), and outside of that range there is little or no gradient in feedback (see fig. 11d). Simply put, there is only a small range of input values for which a change in input results in a change in output. The function removes the evolutionary pathway. This means that the outcome of the optimization could rely heavily on the random initial state. Averaged results of many runs for this configuration show a similar curve as fig. 26j.

6.3 Hypothesis 3

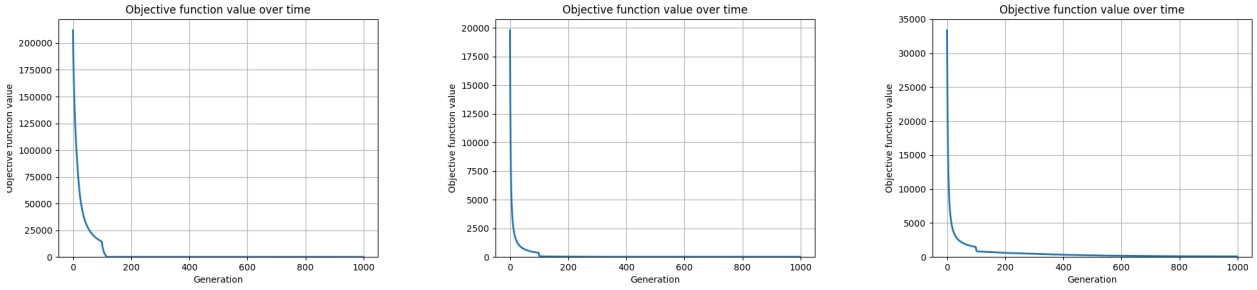
Hypothesis 3 The emerging species converge to a near-optimal decomposition of the problem.

6.3.1 Predator-Prey

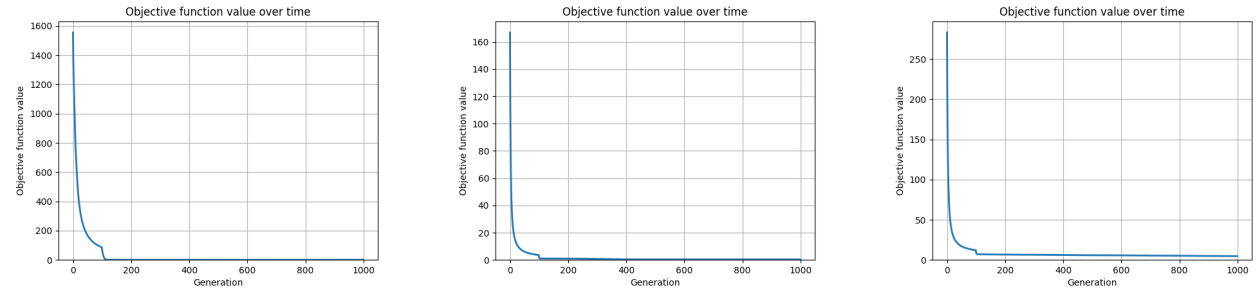
For the Predator-Prey model, it is difficult to determine when the decomposition of the problem is optimal. Most of the time, the model converges to a composition of two distinct species, oscillating between one of one and two of the other (as in fig. 21b). It is possible that a composition of two species is sufficient for a task of this complexity. Figure 23c shows that many compositions of this type are able to achieve a perfect benchmark score of 9/9. On average, compositions of three distinct species evolved separate perform slightly better, as seen in fig. 23a. This could suggest that three distinct species is the optimal decomposition, and the emergent species do not converge to that optimum.

6.3.2 Toxin

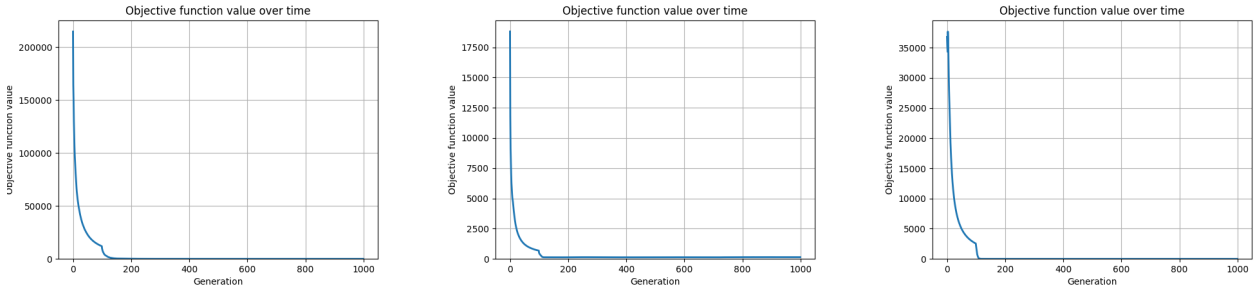
In the Toxin model, the optimal decomposition is more easily determined. An optimal decomposition is one where all costs are minimized – both the cost of the toxin, and the genes. The toxin cost is minimized if there is no toxin left in the environment, as each unit of toxin costs more for the population to take as damage, than it costs a single agent to perform the clean function. The gene cost is minimized by having no redundant functions – if the toxin is completely cleaned, any extra



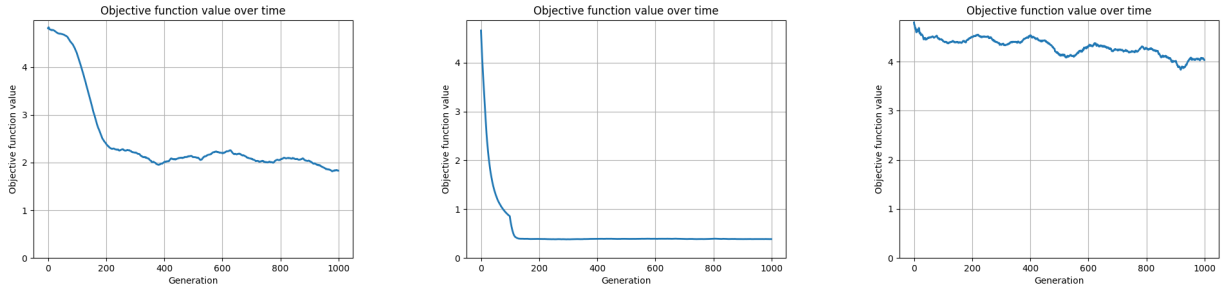
(a) EC, Sphere (b) RC, Sphere (c) FS, Sphere



(d) EC, Griewangk (e) RC, Griewangk (f) FS, Griewangk

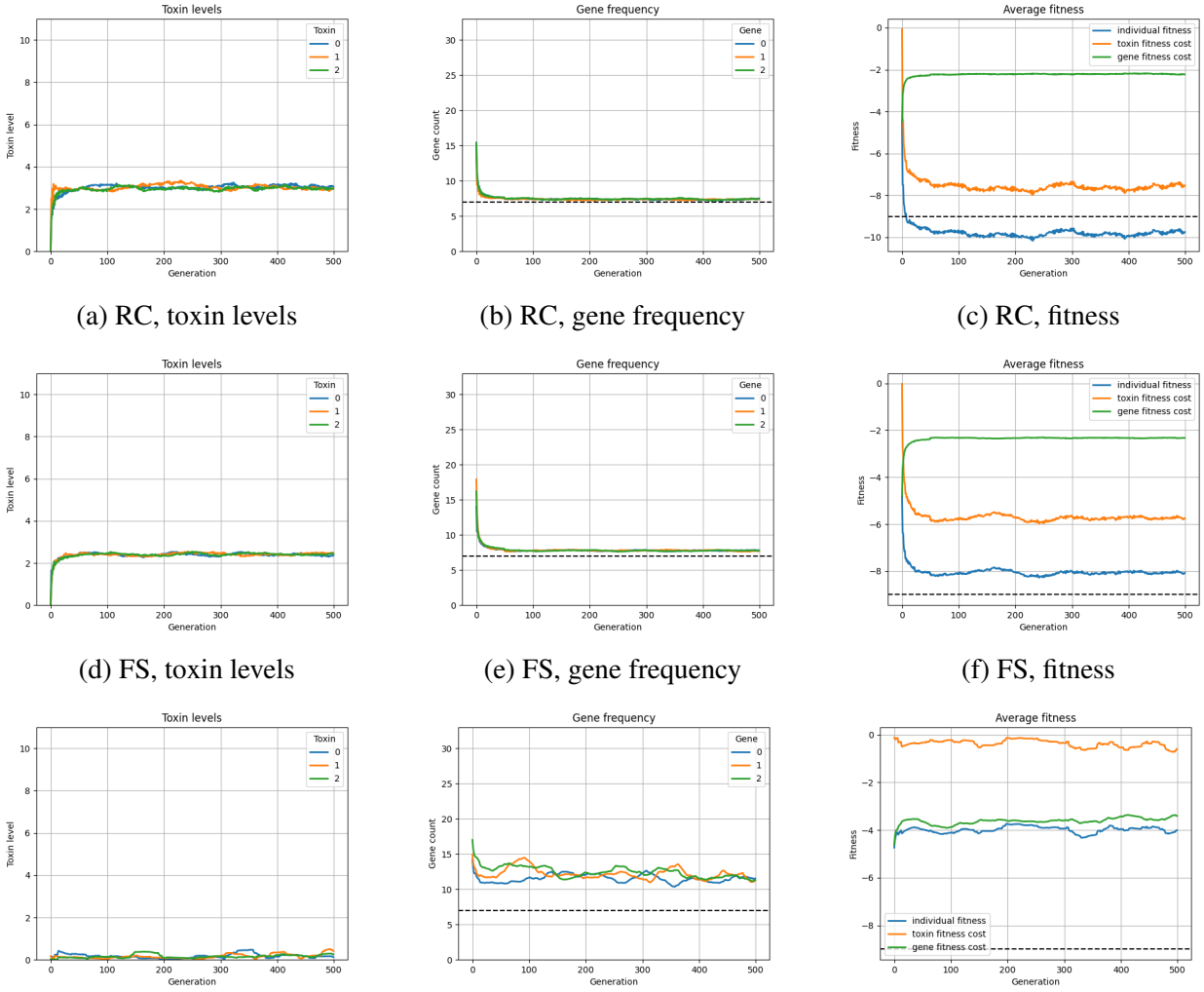


(g) EC, FNS (h) RC, FNS (i) FS, FNS



(j) EC, SPS (k) RC, SPS (l) FS, SPS

Figure 26: Objective value for different model configurations (columns) (Evolutionary Compositions, Random Compositions, Forced Subpopulations) and different optimization functions (rows) (Sphere, Griewangk, Fully Non Separable (FNS), Strongly Partially Separable (SPS)) for the Function Optimization model. The number of dimensions $N = 10$, and for SPS the number of subcomponents $k = 5$.



(a) RC, toxin levels (b) RC, gene frequency (c) RC, fitness

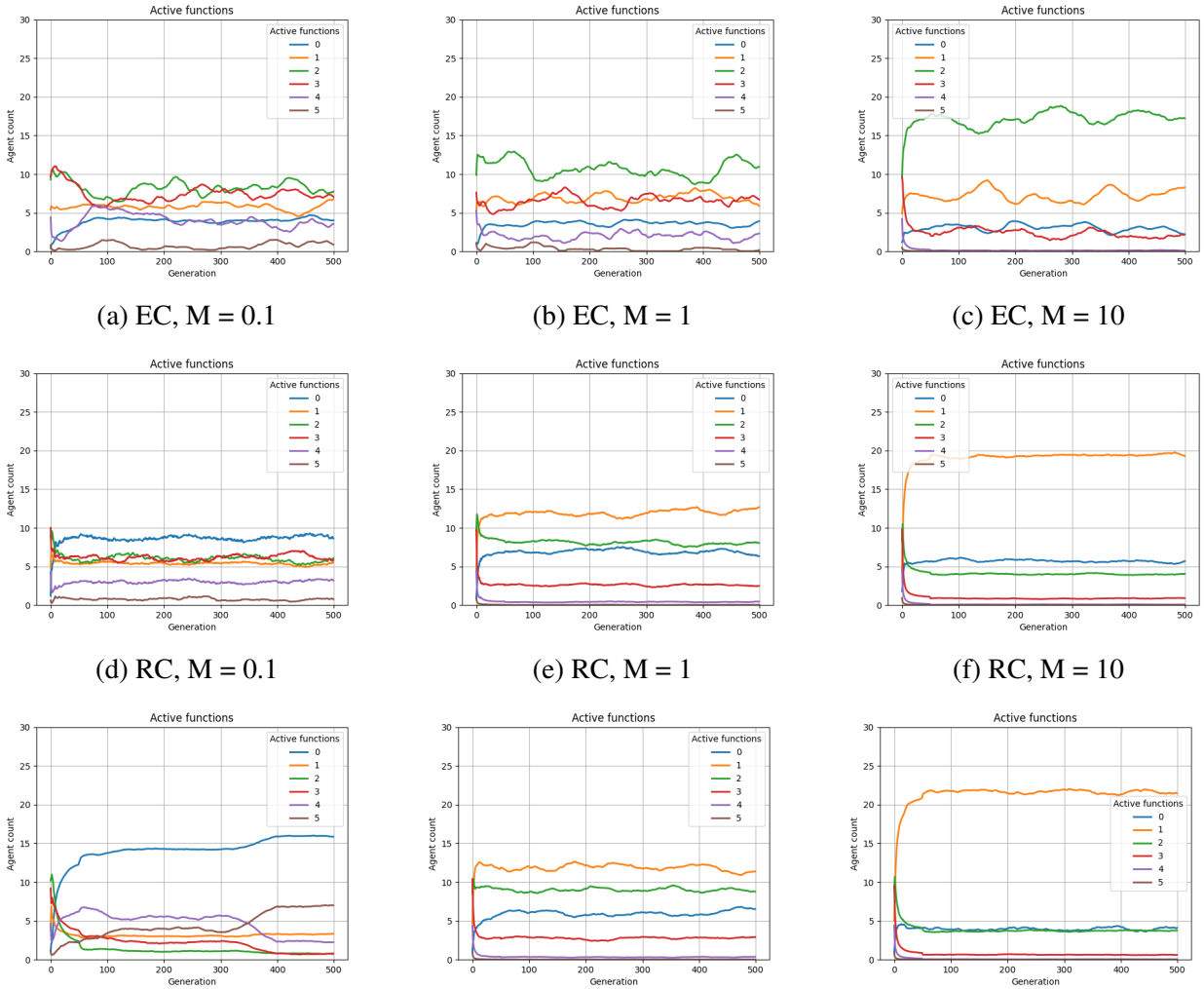
(d) FS, toxin levels (e) FS, gene frequency (f) FS, fitness

(g) EC, toxin levels (h) EC, gene frequency (i) EC, fitness

Figure 27: Optimality of the Toxin model with three different running settings. Left column shows the toxin levels (objective), middle column shows gene frequency (efficiency), right column shows fitness breakdown. The dotted line in the gene frequency graph denotes the Nash equilibrium value. Any value over 10 is redundant, anything under is missing.

cleaning function is wasted energy (see fig. 10b). If the function cost is equal for each agent, function distribution across the population is no factor in optimality. However, if the gene cost multiplier makes subsequent functions cheaper ($M < 1$) or more expensive ($M > 1$), functions should be concentrated or dispersed evenly through species, respectively.

In fig. 28 the distribution of functions among the agent population can be analyzed. Lines show the amount of agent with the corresponding number of active functions. 0 means that an agent has no active functions, 5 means it performs the cleaning function for all present toxins. M is the cost exponent, used in calculating the function cost eq. (13), for $M < 1$ each subsequent function has lower cost, $M > 1$ increases the cost for each extra active function after the first. The most cost efficient distribution of functions for $M < 1$ would be for a proportion of the agent population to adopt all functions, and all others performing no function. For $M > 1$ the most cost efficient distribution is to have the most diffuse possible distribution of function among the agent population. For $M = 1$ there is no selection pressure on distribution of functions, and so the expected distribution should be normal.



(a) EC, M = 0.1 (b) EC, M = 1 (c) EC, M = 10

(d) RC, M = 0.1 (e) RC, M = 1 (f) RC, M = 10

(g) FS, M = 0.1 (h) FS, M = 1 (i) FS, M = 10

Figure 28: Plots showing the distribution of functions among agents in the Toxin model. The y-axis shows the number of agents with the amount of functions active associated with the line. Three settings of the model are run with different function cost multiplier (M) settings. Simulations for $|T| = 5$.

In all three model type runs the same effects can be observed in varying strengths. A lower value for M increases the viability for having multiple active functions, and thereby increasing the proportion of agents without a function. The optimal composition for $M = 10$ would be to maximize the distribution of functions across the agents, and to minimize the highest number of functions on a single agent, in order to minimize the amount of times the multiplier function cost is incurred. For 5 toxins, 10 base toxin, 30 agents and 1 cleaning rate this comes down to 50 active functions total distributed over 30 agents, giving an optimally diffuse composition of 10 agents with 1 function active, and 20 agents with 2 functions active. Figure 28c does come close to this distribution. Given $M = 0.1$ the aim of the distribution is the exact opposite - it should maximize the number of times the cost multiplier is applied to minimize the gene fitness cost on the population. The optimal distribution here is 10 agents with 5 functions active, and 20 agents with 0 functions active. Figure 28a shows that this is not the distribution of functions achieved by the model. The distribution more closely resembles the normal distribution of fig. 28b.

6.3.3 Function Optimization

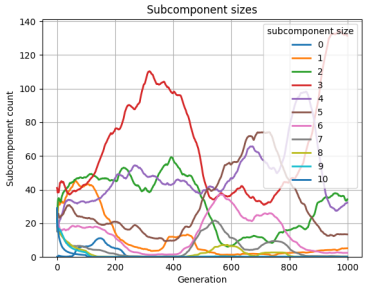
The Function Optimization model can determine optimal decomposition by analyzing the separability of the used optimization function. An optimally decomposed solution is one where each nonseparable subcomponent is grouped in a species. An optimal composition would entail a fully heterogeneous composition of species representing nonseparable subcomponents. Each position of the solution vector would be filled by a gene of at least one species. Every overlap of positions on the vector are explained by the variable being part of multiple nonseparable subcomponents.
The Fully Non-Separable (FNS) function has an optimal subcomponent size of 10 (given $N = 10$), encompassing all 10 variables in a single subcomponent. The Strongly Partially Separable (SPS) function, defined with $N = 10$ and $k = 5$, has 5 subcomponents per composition, leading to an optimal subcomponent size of 2. Thus, the ideal composition sizes are 1 for FNS and 5 for SPS. Note that the composition size for RC and FS is constant at 1 for RC and the number of dimensions (10) for FS.
However, as shown in fig. 29, particularly in fig. 29a and fig. 29g, neither function predominantly exhibits its optimal subcomponent size. SPS's subcomponent sizes appear normally distributed around a mean of 5, indicating minimal evolutionary pressure on subcomponent size. In RC, a convergence towards a stable distribution of subcomponent sizes is observed, intriguingly aligning more closely with the optimal distribution for the alternate function rather than its own.
For composition sizes, neither FNS nor SPS achieves its optimal configuration; FNS's ideal being a single subcomponent of size 10, and SPS's, five subcomponents of size 2 without overlap. The subcomponent and composition size graphs for the forced subpopulation model are included as a reference. Each subcomponent consists of a single variable, and a composition has one component for each variable.

6.4 Hypothesis 4

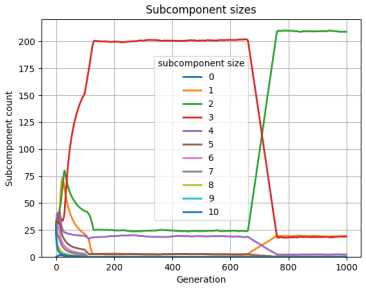
Hypothesis 4 A hybrid approach combining individual and team level selection promotes the survival of self-sacrificing altruistic behaviour in a competitive environment.

6.4.1 Toxin

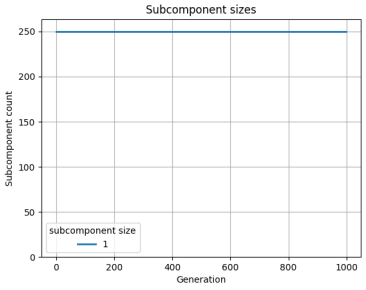
Agents purely interested in maximizing personal fitness can lead to a sub-optimal Nash equilibrium state. This is most clearly seen in the Toxin model, as described in section 4.2.2. In the Toxin model,



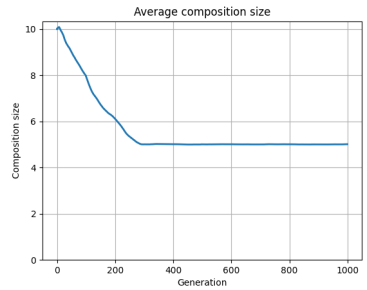
(a) EC, FNS, subcomponent size.



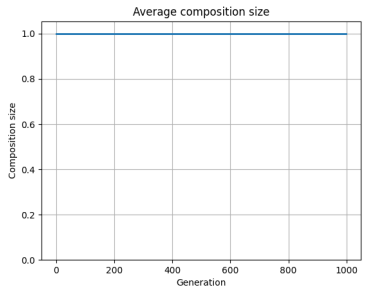
(b) RC, FNS, subcomponent size.



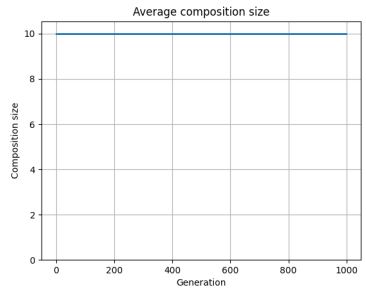
(c) FS, FNS, subcomponent size.



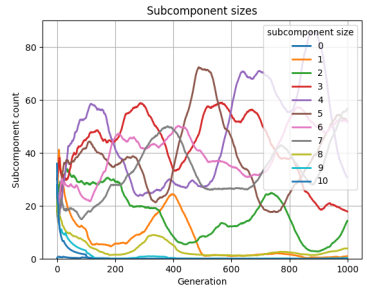
(d) EC, FNS, composition size.



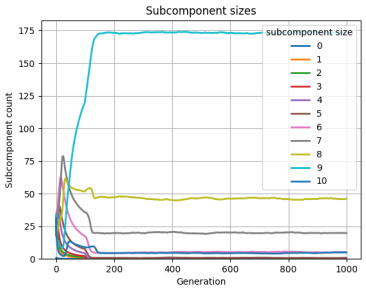
(e) RC, FNS, composition size.



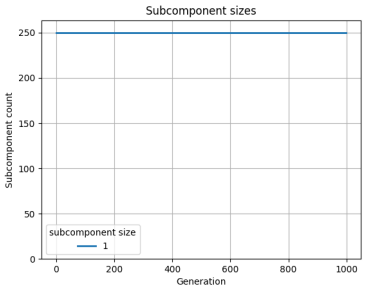
(f) FS, FNS, composition size.



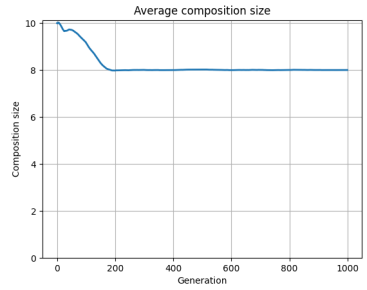
(g) EC, SPS, subcomponent size.



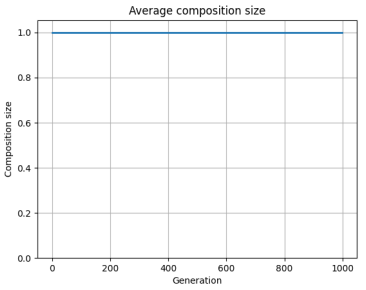
(h) RC, SPS, subcomponent size.



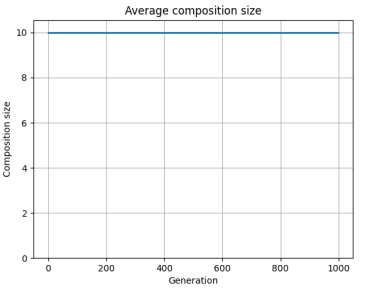
(i) FS, SPS, subcomponent size.



(j) EC, SPS, composition size.



(k) RC, SPS, composition size.



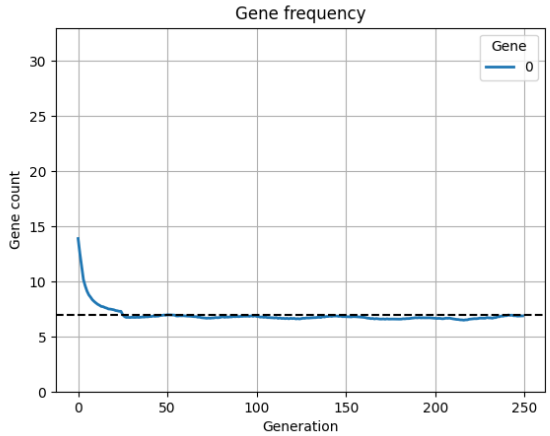
(l) FS, SPS, composition size.

Figure 29: Average composition and subcomponent sizes for different model configurations and optimization functions for the Function Optimization model. The subcomponent size is the number of non-knocked out genes on a chromosome, which are the number of genes contributing to the objective solution. The composition size is the number of chromosomes (subcomponents) included in a final evaluation.

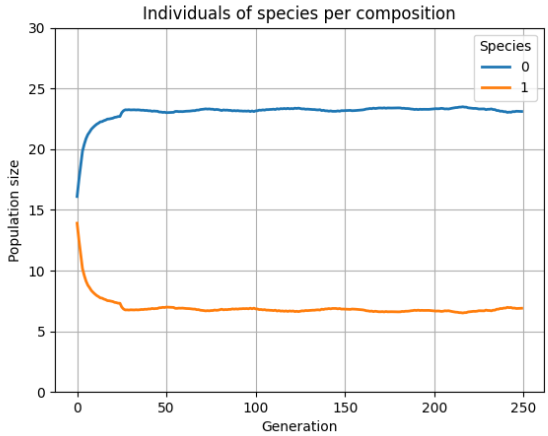
to achieve the maximal social welfare Pareto-optimum, it is required that some agents engage in self-sacrificing altruistic behaviour by adopting the cleaning function at net personal cost. The cost to the individual yields a substantially larger net gain for the population.
The species representation fitness scaling detailed in section 4.3.8 introduces a team level selection mechanic that should allow the altruistic species to survive. Evolutionary compositions are required for this technique to have any effect. Applying representation scaling to a model with random agent sampling has no effect, since random agent sampling gives compositions with distributions that mirror the population distribution.
As baseline we take the Toxin model with random sampling and personal fitness fully determined by the individual fitness (eq. (15), with $cfw = 0$ in eq. (33)) fig. 30. As expected, the number of agents with the cleaning function moves to the Nash equilibrium. Experiments with EC and fitness scaling (fig. 31) show that the number of agents with the cleaning function moves to the Pareto-optimal number, thereby achieving the maximal social welfare state. The optimal composition with 10 helper agents and 20 beneficiaries is stable. The proportional difference between the composition distribution (fig. 31b) and population distribution (fig. 31e) shown in fig. 31f confirms that altruistic species are at a disadvantage in the agent population and at an advantage in the compositions. Note the simulations here are run with an agent population of 30. The results are the same at larger scale, but these parameters produce a clearer view.
The same result can be achieved by setting the personal fitness fully to the collective fitness ($cfw = 1$), the average fitness of all composition evaluations it participated in (see section 3.5.1), as seen in fig. 32. Here, individuals are selected for only based on team performance. This simulation shows that without individual fitness, he fitness of each species is about equal fig. 32c.

6.4.2 Predator-Prey

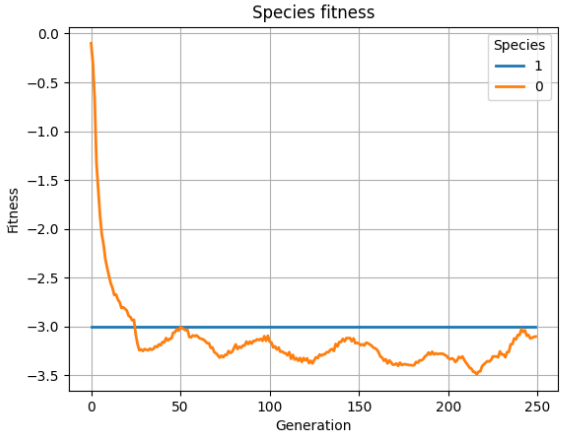
These result do not extend to the Predator-Prey model. Figure 33 show the results obtained when the personal fitness is wholly determined by the collective fitness ($cfw = 1$), when applied to the Predator-Prey model. In fig. 33d it is visible a three species composition emerges around generation 20. The composition steadily increases in performance for 20 some generations as visible in fig. 33a and fig. 33b. Then, one of the three species (species 92) drops in number and goes extinct around generation 50, as seen in fig. 32e. The two remaining species are unable to recover performance, and the species slightly more adapted to the new environment (218) out competes and consequently crowds out the remaining other species (0), resulting in a fully homogeneous population around generation 90. Species 218 recovers some performance as it adapts to a fully homogeneous team environment.



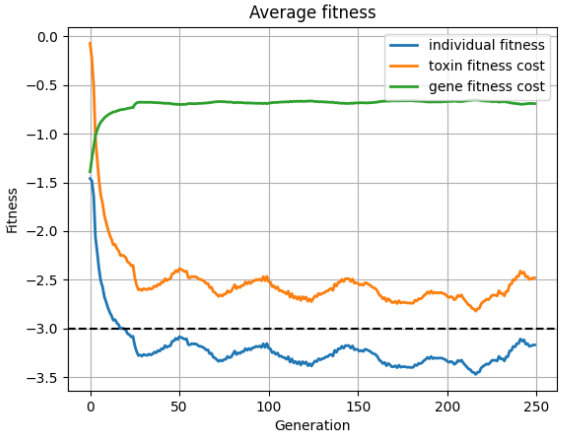
(a) RC, active cleaning function prevalence.



(b) RC, average species distribution per composition.

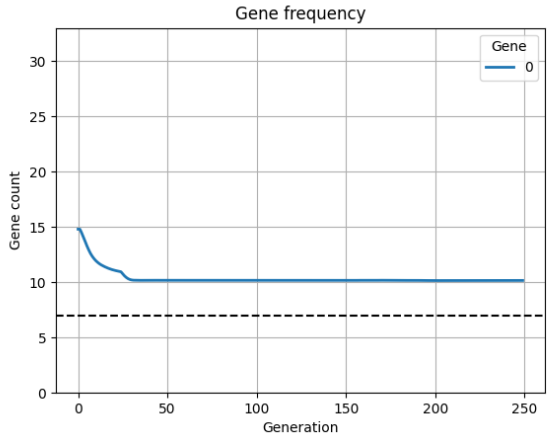


(c) RC, average fitness per species.

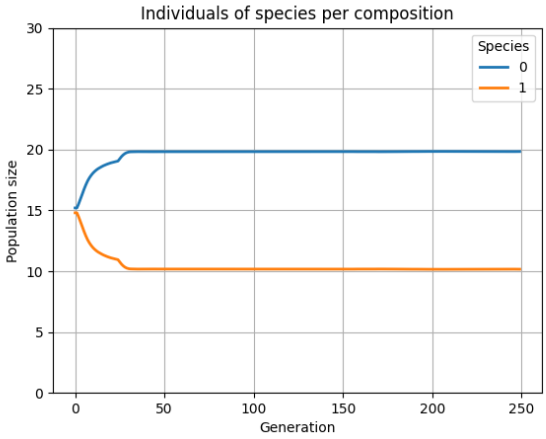


(d) RC, average composition of the fitness calculation.

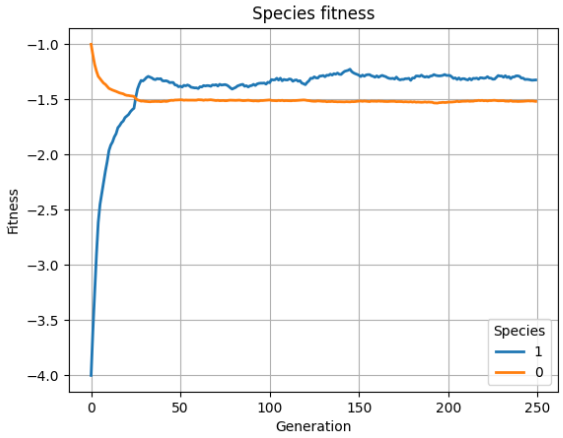
Figure 30: Toxin model random compositions baseline, showing the model moves to a sub-optimal Nash equilibrium. Random sampling for the compositions means that the species per composition (fig. 30b) shows the species population distribution.



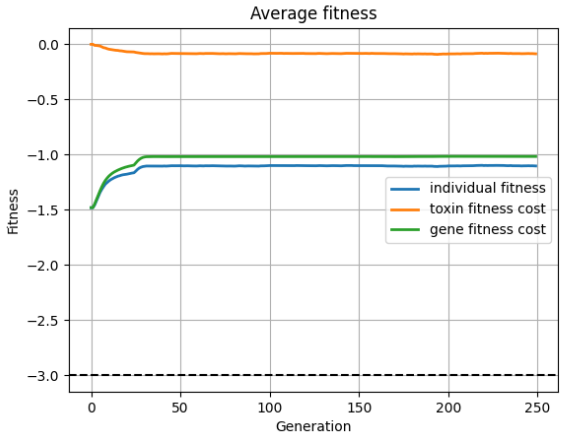
(a) EC, active cleaning function prevalence.



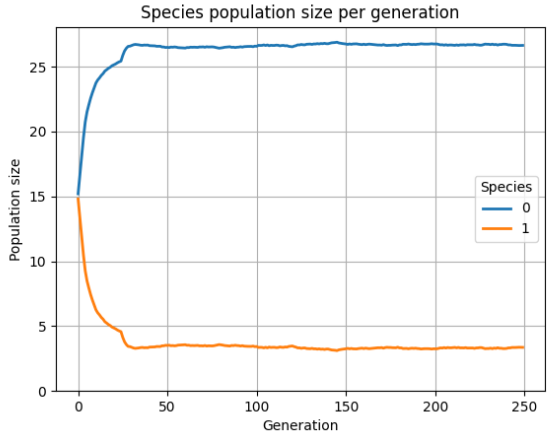
(b) EC, average species distribution per composition.



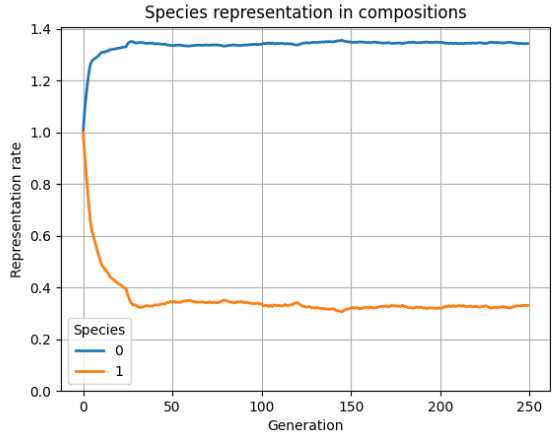
(c) EC, average fitness per species.



(d) EC, individual fitness.

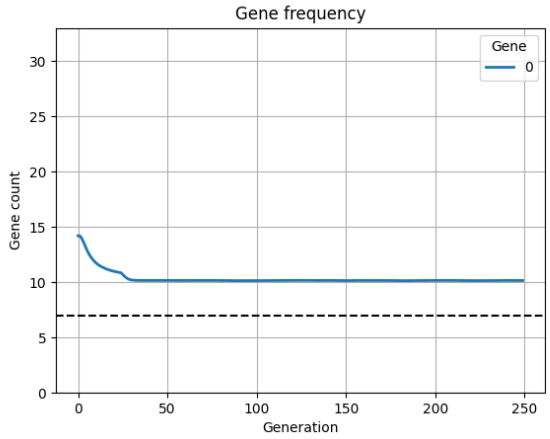


(e) EC, species population distribution.

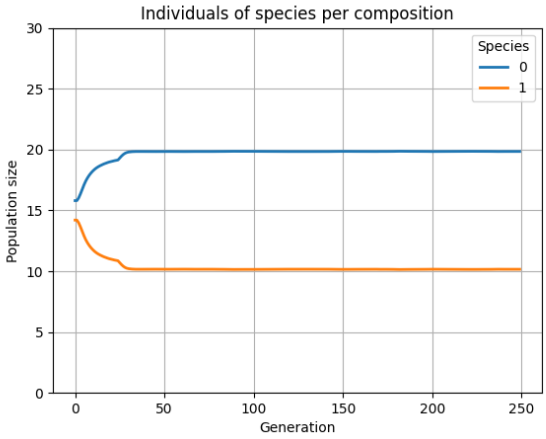


(f) EC, representation rate of species in compositions.

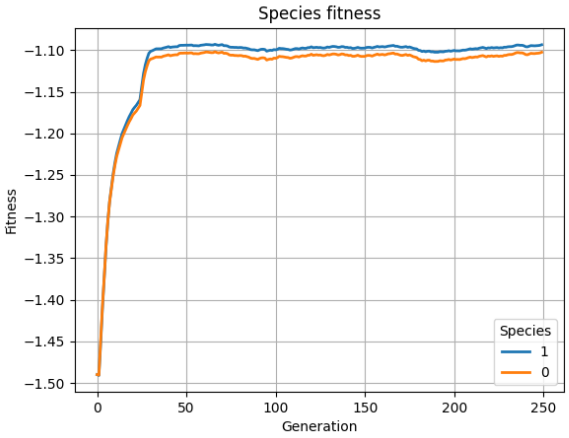
Figure 31: Toxin model evolutionary compositions with fitness scaling, showing the model moves to the Pareto-efficient maximal social welfare.



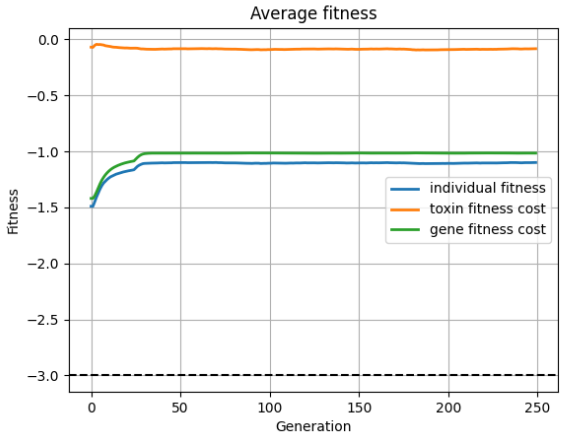
(a) EC, gene frequency.



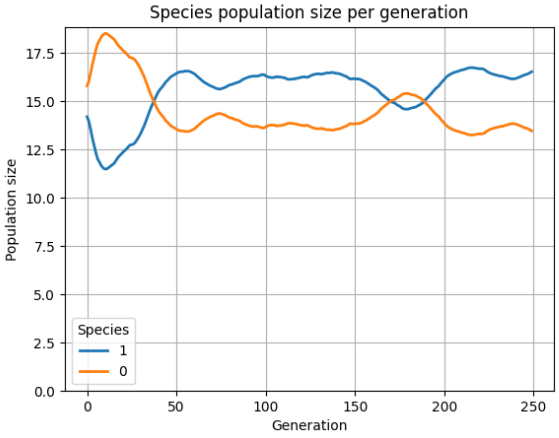
(b) EC, average species distribution per composition.



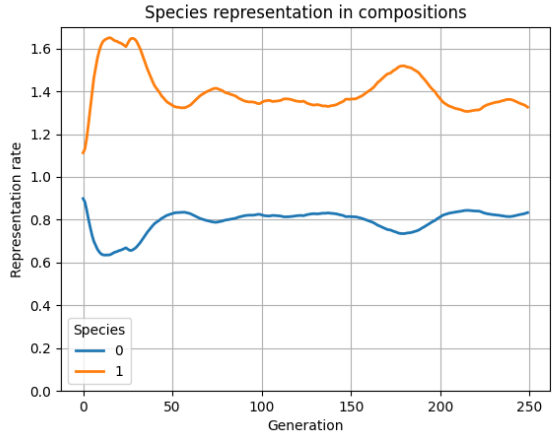
(c) EC, species fitness.



(d) EC, individual fitness.

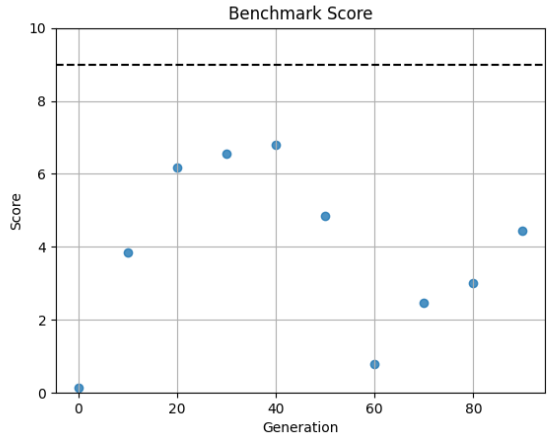


(e) EC, species population distribution.

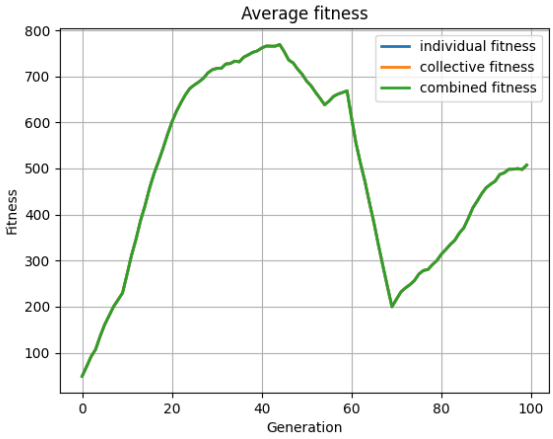


(f) EC, representation rate of species in compositions.

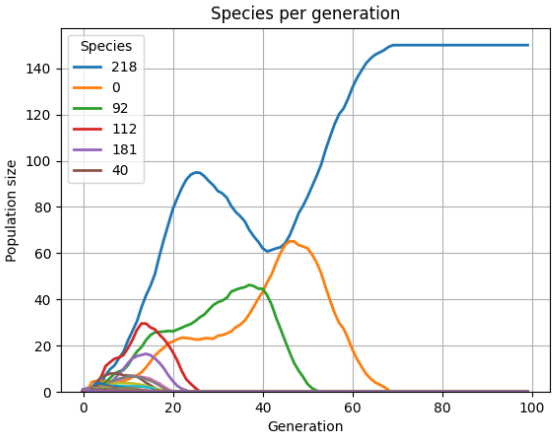
Figure 32: Toxin model evolutionary compositions with individual fitness fully determined by team performance. Without species representation fitness scaling.



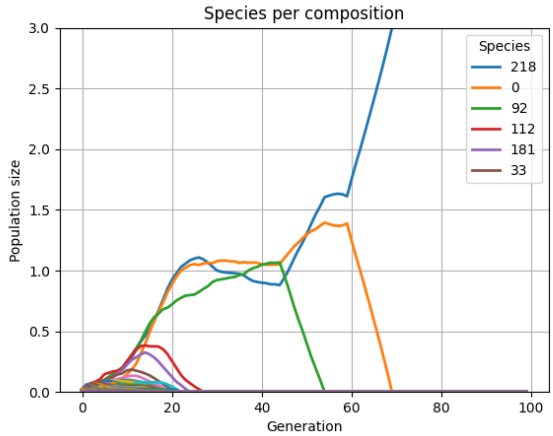
(a) EC, benchmark.



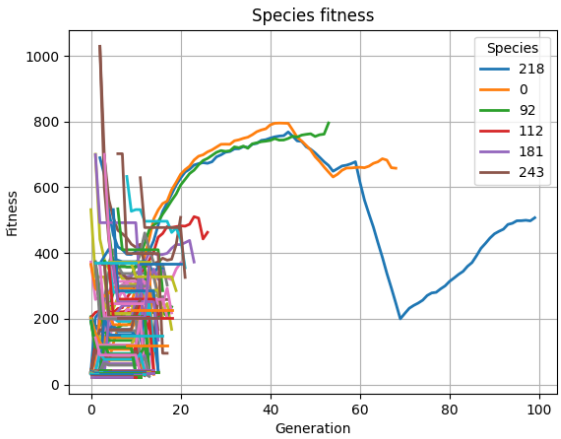
(b) EC, average fitness per agent.



(c) EC, agent species population distribution.



(d) EC, agent species compositions distribution.



(e) EC, species fitness.

Figure 33: Predator-prey model evolutionary compositions with individual fitness fully determined by team performance. Without species representation fitness scaling.

7 Discussion

This section discusses the results in light of the research question and the theory. The value of the novel mechanics is reviewed briefly, some limitations of the model are discussed as well as some suggested areas where improvements and refinements can be made. Finally, the work is summarized and concluded.

7.1 Interpretation of results

Hypotheses The results show that the first hypothesis can be confirmed by observation. The results in section 6.1 clearly show that multiple species can emerge and coexist stably within the same population. The difference in cluster shape between fig. 22c and fig. 22b and the similarity to fig. 22a show the compartmentalization and clustering effect of genome distance based mating. The second hypothesis can also be confirmed. Individuals from multiple species can form a composition that is a full solution to the problem, even if these species are evolved in the same population. Species coevolve as an interdependent composition. Performance is higher when the composition is functionally heterogeneous. Emergent species compositions performance is not consistently better than forced subpopulations, and at times worse. The third hypothesis can not be confirmed. While species form a decomposition of the problem, and finding the optimal solution is difficult, the results do not show consistent near-optimal decomposition of the problem. The fourth hypothesis can be confirmed. Self-sacrificing behaviour can be allowed to thrive in a competitive environment if there is some form of team level selection in combination with individual level selection.

Variable grouping One possible reason for the failure to produce a near-optimal decomposition could lie in that there is too little or no evolutionary pressure for linked variables to self-organize in subcomponents. Reasoning for the optimization model, whether the variables of a subcomponent of the function are grouped in the same chromosome species needs to confer an advantage to the species over any other configuration. A possible fitness advantage, though very weak, is that a correct grouping of linked variables without any other non-linked variable gives a more stable fitness peak – any mutation within such a species gives a predictable and consistent resulting change in fitness. The strength of this fitness advantage is hard to determine, and could easily be outweighed by the fitness penalty applied for each subsequent variable in a species, which is intended to pressure variables to only group when they are linked.

Toxin optimization In the Toxin model this fitness advantage to self-organize in optimal groups is forced by adjusting the M cost multiplier parameter. This cost multiplier changes the Pareto-optimal and maximal social welfare distribution. Analyzing why for $M = 10$ the population does organize nearing an optimal composition and why it does not for $M = 0.1$ provides insights in the specific dynamics at play. One partial explanation is that for $M = 10$ the fitness of species diverges strongly and so has a strong evolutionary effect. In contrast, for $M = 0.1$ any extra application of the cost multiplier gives diminishing returns; the fitness difference between a 4 function species and a 5 function species is insignificant. While this dynamic could play a role, it is an incomplete explanation. The more important dynamic for this result is to realise that once the collective fitness objective is satisfied, in this case cleaning all the toxin from the environment, constituents are free to optimize their own fitness. With $M = 0.1$, the 'optimal' composition that minimizes both toxin and gene cost features 10 agents with 5 functions, and 20 with 0. This is not achieved (fig. 28a), because increasing the number of functions for an individual agent does not confer any benefit to the individual. In this

configuration the collective fitness and so the fitness of a composition is set to the remaining toxin in the environment. Because of this, compositions are agnostic to the distribution of functions, as long as the objective is met. It also means that there is no evolutionary pressure from team selection to reduce the number of functions past saturation. So the self-sacrificing behaviour (of adopting more functions at a discount to save on total function cost) is not compensated for with fitness scaling. The total function cost is never a metric for which is optimized. Then, why does $M = 10$ approach the optimal composition of 20 2 function agents and 10 1 function (fig. 28c)? Because they all try to minimize their own function cost and thus number of functions, and not doing so is much less viable. For $M = 1$ it would also be expected that over time the agents minimize the number of functions per agent, the results (fig. 28b) show this less than expected. This could be because many species are and remain equally viable, just as many compositions are equally viable, resulting in that the compositions never converge. This means the group of agents with no functions is always large (for having the lowest function cost), which gives a selection advantage (through fitness scaling) to agents that produce successful compositions that include many agents with no function. The conclusion we can draw from this is that if a collective value needs to be optimized, it should be included in the objective function for the compositions. Experiments (in appendix C) that set the collective fitness to the average fitness of all, thereby including both the toxin and function cost, confirm this. As an interesting aside point, the inability for functions to converge on a single species, even if at great discount for each subsequent function, could be considered an argument against the 'shooting the moon' theory of the BQH [18], where a species that has become a helper for one function is more likely to become helper for other functions, becoming a 'keystone species' as evolutionary strategy.

Collective versus hybrid The results of fig. 32 lead to an interesting question: if a collective fitness for individuals leads to the system approaching the maximal social welfare state, why bother with individual fitness and fitness scaling at all? In a forced subpopulations approach, this strategy is proven to work. Theoretically, there is still an incentive for agents to optimize and out compete competitors if they contribute more to the collective, including if that is done by specializing and becoming more efficient personally. The problem lies with the emergent species approach. Results of this collective fitness only scenario in the Predator-Prey model show in fig. 33 that performance increases for some generations, after which it declines sharply. It declines because one or more obligate mutualistic species have gone extinct. Why they go extinct exactly is difficult to determine, but one explanation could be that the marginally worse performing species is crowded out by the other(s). As fig. 32c and fig. 33e show, species' fitness are close in value. This is advantageous in the sense that species average to the same population sizes as fig. 32e shows, which gives them equal opportunity for evolution. However, it does not prevent a species that marginally out competes others from crowding out the species it relies on. This is not a problem in the Toxin model (fig. 32) as species are only a few mutations away and so are effectively never 'lost'. But when the sequence space for species is very large, extinction of a species means a lineage is lost and will not re-emerge easily. Simply put, there needs to be correction mechanism preventing species useful to compositions from going extinct, and the fitness scaling does exactly that. It allows a population of individualist self-interested reward maximizing agents to produce the same Pareto-optimal results as a population of collectivist agents, with the additional benefit of conserving necessary diversity.

Function distribution One of the objectives of the thesis is dynamic task allocation. It is difficult to conclude if that is achieved. Tasks in the Toxin model are atomic, tasks in the Predator-Prey model are ambiguous and no thorough behavioural analysis is performed on predator species strategies, and in the Function Optimization model while 'correct' variable grouping is preferred, any variable group-

ing resulting in a heterogeneous solution is dynamic task allocation. 'Tasks' insofar as they can be discretized might not necessarily assort in species as the path of least resistance. The idea is that multiple multi-function species emerge, and through black queen effects functions are transferred, lost, and converged in species such that functions are unique per species. This effect is not demonstrably observed from the experiments. Speciation and niching occur for as far as it is evolutionary advantageous to do so. If theoretically it is advantageous to always have a jack-of-all-trades within the ranks, it is possible for that to emerge. If it is advantageous for some type to niche completely and lose all other functions, because it can rely on others for that task, that can also happen. Clean compartmentalization and modularity of function might simply not be a common evolutionary strategy. We find some evidence for this conclusion if we look at the functionality of the brain, enzymes or of genes. In this light, dynamic task allocation might simply not result in a distinct and recognizable distribution of functions.

7.2 Novel mechanics review

This section briefly summarizes the contributions of the introduced novel mechanisms.

Genome Distance Based Mating The genome distance based mating has the intended effect of converging candidate solutions to multiple areas in the sequence space. Hypothesis 1 and 1.1 confirm this. It is not the scope of this thesis to asses whether it alone works well as a general method of preserving diversity in a single population. For the purpose of allowing multiple species to emerge the mechanic works as intended, though still rudimentary in implementation. There are still many gains to be made from the concept, which are discussed in section 7.4.

Evolutionary Compositions The primary objective of the evolutionary compositions is sampling in a way that allows symbiotic relationships to form through coevolution. The secondary objective that follows from the first is to produce good compositions. By confirming hypothesis 1, 2 and 2.1 the conclusion can be drawn that both objectives are achieved. The technique explores the search space of compositions with more informed direction than another sampling method.

Fitness Scaling Species proportional fitness scaling is intended to help species that are good for the collective but are at a disadvantage individually to survive. Hypothesis 4 shows that this goal is achieved by utilizing this mechanism.

7.3 Limitations

Potter in his dissertation[14] mentions that all single-gene pool approaches are limited to the evolution of subcomponents that share a common representation. This is true for the current implementation.

Scalability Questions can be raised over the scalability of the model. The thesis has opted to favour testing the model in different domains over deep performance runs. All tested domains have been simple or abstracted versions of more real world applications. The Predator-Prey model only requires a maximum of three different species, the Function Optimization model is only tested with a low number of dimensions whereas the challenge is often in the high number of dimensions, and the Toxin model is an abstract model with clear dynamics. The model, by making the (necessary) adaptation to a single genome pool, loses a distinct advantage that traditional CCEAs have of parallel optimization

of subcomponents. On top of that, it increases the number of (sometimes costly) fitness evaluations required by evaluating different compositions of species, whereas with forced subpopulations only one composition of species is ever evaluated. The extra steps taken in genome distance based mating and species detection require many high dimensional distance calculations that scale non-linearly as population size increases. It is clear that the proposed model is not an improvement with regard to computation time.
Other questions remain over the scalability of the functionality. What is the role of population size? The population size is a clear constraint on the number of species that can develop in parallel, and it could be true that a population size that is too small does not allow the required or optimal number of species to emerge. It is also not clear what challenges emerge when the number of required species is large, and if the current model implementation is capable of handling those challenges.

Species Detection and Tracking limitations Species detection and tracking are tasks where perfection is not possible because of the inherent ambiguity of identity. Unsupervised cluster detection features many different trade-offs, and what should be classified as species is equally subject to trade-offs. Balancing these demands can be tricky. The ε value for DBSCAN is important to set such that it accurately distinguish different species. The optimal value for ε changes with the characteristics of the population, such as the population size or dimensions of the agent chromosome. E.g., if the population size is higher, the density of the data increases, and it becomes more likely different clusters are connected though some intermediate points, making a lower ε value more suitable. Improvements to species tracking can be made by tracking density peaks or other more accurate cluster prototypes than the currently used centroid prototype method. This would make more sense as cluster detection is performed with a density based model (DBSCAN). The current centroid method is implemented because centroids work as a heuristic, and are much faster and easier to calculate than density peaks.

7.4 Future work

The current implementation of the genome distance based mate selection mechanic is quite primitive. Selecting a sample from the population to find the closest neighbour in solves some technical issues and gives some control over the granularity of the search, but there is still a lot of dormant undeveloped potential in the mechanic. One alternative mode for genome distance based mate selection is to select a random mate based on species. An explicit (tunable) exploitation/exploration parameter can determine if a mate is selected from the own species, or from another. This gives a closer resemblance to the island model. A problem with this mode for the current model implementation is that it would be ineffective early on, when many different species are recognized and comprise of only a single individual. Another alternative is distance *proportional* mate selection. Converting distances to likelihoods such that genomes close by are selected proportionally more than those at a distance. Introducing a scalar determining the relative weight of distance in this likelihood calculation gives control in the same way as the sample size does in the current implementation. Furthermore, this scalar can be adjusted using the local density of the cluster of the mate selecting genome to dynamically scale exploitation and exploration based on local topological requirements. A third extension takes inspiration from biological life. In nature, population diversity is necessary, and mating with the genomic nearest neighbours (family) has a risk of causing detrimental inbreeding effects. Similarly, mating with genomically very distant mates risks losing specialization benefits or producing unviable offspring. This suggests that there is an optimal range (without making claims over the variability of or distribution within that range) between minimizing and maximizing genomic distance between

mates. Further study to the optimal distance range can be done.

Genome distance based mating and fitness scaling are methods aimed to promote speciation and conserve diversity. Providing a fitness benefit based on the level of diversity in composition chromosomes could increase diversity and promote speciation even more. Regarding the fitness function as objective function, applying a penalty to low-diversity composition chromosomes gives a fitness advantage to high diversity composition solutions. Similarly, a form of novelty search giving a fitness advantage to agent chromosomes that are more unique promotes novel solutions, and thereby exploration for new niches.
The current implementation of species representation fitness scaling, similarly to genome distance based mating, is a platform with more development potential. The mechanic introduces a negative feedback loop that prevents important species from going extinct by utilizing the difference between relative species occurrences in the individual and composition populations. A simple normalized representation ratio scalar proved to be effective. Replacing this scalar with a function could expand functionality to include not only the scalar, but a tuning parameter, a simulated-annealing like temperature or other context metrics to give more control over the strength of the effect. Increasing the effect contributes to dynamic resource allocation by equalizing the proportion of species across populations. Thereby, a species has a population size (evolutionary resources) closer to how much the species is utilized in solutions. Right now the altruist species is kept alive but still lags behind in population size, giving it a stronger fitness advantage pushes the equilibrium ratio closer to 1. Decreasing the effect can be useful if a more homogeneous final solution is desired.

7.5 Conclusion

This thesis explores applying the hologenome theory of evolution within computational evolution, aiming to create functionally heterogeneous compositional 'chimera' entities through coevolution. It leverages emergence and self-organization directed by evolutionary pressures in combination with providing the capacity potential for desirable solutions to emerge. The method uses a single gene pool from which a subset is sampled to form a solution composition. Compositions are composed of any number of individuals from any number of species. A species is a collection of similar chromosomes in the gene pool labeled using an unsupervised clustering algorithm. Speciation and species persistence are enabled by genome distance based recombination, functioning as permeable compartmentalization within the gene pool to produce parapatric speciation. Solution compositions are coevolved with the gene pool of individuals, and through them team-level selection is applied to individuals, promoting both good individual solutions as solutions that contribute to the collective.
The introduction of three novel mechanics; genome distance based mate selection, evolutionary compositions and species proportional fitness scaling, in combination with the application of unsupervised cluster detection to assign species, to the framework of cooperative coevolution is sufficient to achieve emergent symbiotic speciation. Showing only modest performance improvements over over existing algorithms, the model functions as proof-of-concept showcasing the design philosophy of emergent symbiotic speciation for hologenome formation.

Bibliography

[1] G. Rozenberg, T. Bäck, and J. N. Kok, eds., *Handbook of Natural Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[2] S. R. Bordenstein and K. R. Theis, "Host Biology in Light of the Microbiome: Ten Principles of Holobionts and Hologenomes," *PLOS Biology*, vol. 13, p. e1002226, Aug. 2015.

[3] E. Rosenberg and I. Zilber-Rosenberg, *The Hologenome Concept: Human, Animal and Plant Microbiota*. Cham: Springer International Publishing, 2013.

[4] E. Rosenberg and I. Zilber-Rosenberg, "Symbiosis and development: the hologenome concept," *Birth Defects Research Part C: Embryo Today: Reviews*, vol. 93, no. 1, pp. 56–66, 2011.

[5] M. Youle, N. Knowlton, F. Rohwer, J. Gordon, and D. A. Relman, "Superorganisms and Holobionts: Looking for a term for the functional entity formed by a macrobe and its associated symbiotic microbes and viruses? The term is "holobiont"," *Microbe Magazine*, vol. 8, pp. 152–153, Apr. 2013.

[6] U. Kutschera, "Systems biology of eukaryotic superorganisms and the holobiont concept," *Theory in Biosciences*, vol. 137, pp. 117–131, Nov. 2018.

[7] X. Ma, X. Li, Q. Zhang, K. Tang, Z. Liang, W. Xie, and Z. Zhu, "A Survey on Cooperative Co-Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 23, pp. 421–441, June 2019.

[8] M. Meselhi, R. Sarker, D. Essam, and S. Elsayed, "A decomposition approach for large-scale non-separable optimization problems," *Applied Soft Computing*, vol. 115, p. 108168, Jan. 2022.

[9] D. Sofge, K. De Jong, and A. Schultz, "A blended population approach to cooperative coevolution for decomposition of complex problems," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, vol. 1, (Honolulu, HI, USA), pp. 413–418, IEEE, 2002.

[10] G. A. Trunfio, "A Cooperative Coevolutionary Differential Evolution Algorithm with Adaptive Subcomponents," *Procedia Computer Science*, vol. 51, pp. 834–844, 2015.

[11] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative Co-Evolution With Differential Grouping for Large Scale Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, pp. 378–393, June 2014.

[12] M. N. Omidvar, Y. Mei, and X. Li, "Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, (Beijing, China), pp. 1305–1312, IEEE, July 2014.

[13] M. A. Potter and K. A. D. Jong, "Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents," *Evolutionary Computation*, vol. 8, pp. 1–29, Mar. 2000.

[14] M. A. Potter, *The design and analysis of a computational model of cooperative coevolution*. George Mason University, 1997.

[15] A. Byrski, R. Dreżewski, L. Siwik, and M. Kisiel-Dorohinicki, "Evolutionary multi-agent systems," *The Knowledge Engineering Review*, vol. 30, pp. 171–186, Mar. 2015.

[16] L. Panait and S. Luke, "Cooperative Multi-Agent Learning: The State of the Art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, pp. 387–434, Nov. 2005.

[17] S. Wright, "Isolation by distance," *Genetics*, vol. 28, no. 2, p. 114, 1943.

[18] J. J. Morris, R. E. Lenski, and E. R. Zinser, "The Black Queen Hypothesis: Evolution of Dependencies through Adaptive Gene Loss," *mBio*, vol. 3, pp. e00036–12, May 2012.

[19] A. Mas, S. Jamshidi, Y. Lagadeuc, D. Eveillard, and P. Vandenkoornhuyse, "Beyond the Black Queen Hypothesis," *The ISME Journal*, vol. 10, pp. 2085–2091, Sept. 2016.

[20] M. Morange, *The misunderstood gene*. Harvard University Press, 2001.

[21] R. Dawkins, *The selfish gene*. Oxford university press, 2016.

[22] O. Rössler, "Recursive evolution," *Biosystems*, vol. 11, pp. 193–199, Aug. 1979.

[23] D. A. Powers, T. Lauerman, D. Crawford, and L. DiMichele, "Genetic mechanisms for adapting to a changing environment," *Annual Review of Genetics*, vol. 25, no. 1, pp. 629–660, 1991.

[24] T. A. O'Shea-Wheller, E. R. Hunt, and T. Sasaki, "Functional Heterogeneity in Superorganisms: Emerging Trends and Concepts," *Annals of the Entomological Society of America*, vol. 114, pp. 562–574, Sept. 2021.

[25] J. J. Morris, "Black Queen evolution: the role of leakiness in structuring microbial communities," *Trends in Genetics*, vol. 31, pp. 475–482, Aug. 2015.

[26] J. J. Morris and E. Schniter, "Black Queen markets: commensalism, dependency, and the evolution of cooperative specialization in human society," *Journal of Bioeconomics*, vol. 20, pp. 69–105, Apr. 2018.

[27] S. J. Giovannoni, J. Cameron Thrash, and B. Temperton, "Implications of streamlining theory for microbial ecology," *The ISME Journal*, vol. 8, pp. 1553–1565, Aug. 2014.

[28] Y. I. Wolf and E. V. Koonin, "Genome reduction as the dominant mode of evolution," *BioEssays*, vol. 35, pp. 829–837, Sept. 2013.

[29] C. Darwin, *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray, 1st ed., 1859. Original work published 1859.

[30] P. Adamski, M. Eleveld, A. Sood, Kun, A. Szilágyi, T. Czárán, E. Szathmáry, and S. Otto, "From self-replication to replicator systems en route to de novo life," *Nature Reviews Chemistry*, vol. 4, pp. 386–403, July 2020.

[31] M. J. Eleveld, "Departure from randomness: expanding the structure space of self-replicating molecules," 2023.

[32] L. E. Orgel, "The origin of life on the earth," *Scientific American*, vol. 271, no. 4, pp. 76–83, 1994.

[33] N. Eldredge, S. J. Gould, *et al.*, "Punctuated equilibria: an alternative to phyletic gradualism," *Models in paleobiology*, vol. 82, p. 115, 1972.

[34] M. A. Nowak, *Evolutionary dynamics*. Harvard university press, 2006.

[35] L. Cook and I. J. Saccheri, "The peppered moth and industrial melanism: evolution of a natural selection case study," *Heredity*, vol. 110, no. 3, pp. 207–212, 2013.

[36] M. E. Majerus, "Industrial melanism in the peppered moth, biston betularia: an excellent teaching example of darwinian evolution in action," *Evolution: Education and Outreach*, vol. 2, no. 1, pp. 63–74, 2009.

[37] A. Tchaikovsky, *Children of ruin*. Pan Macmillan, 2019.

[38] R. Solé, "Revisiting Leigh Van Valen's "A New Evolutionary Law" (1973)," *Biological Theory*, vol. 17, pp. 120–125, June 2022.

[39] R. Dawkins, *Flights of Fancy: Defying Gravity by Design and Evolution*. Bloomsbury Publishing, 2021.

[40] N. Rønsted, G. D. Weiblen, J. M. Cook, N. Salamin, C. A. Machado, and V. Savolainen, "60 million years of co-divergence in the fig–wasp symbiosis," *Proceedings of the Royal Society B: Biological Sciences*, vol. 272, no. 1581, pp. 2593–2599, 2005.

[41] J.-H. Xiao, Z. Yue, L.-Y. Jia, X.-H. Yang, L.-H. Niu, Z. Wang, P. Zhang, B.-F. Sun, S.-M. He, Z. Li, *et al.*, "Obligate mutualism within a host drives the extreme specialization of a fig wasp genome," *Genome biology*, vol. 14, pp. 1–18, 2013.

[42] M. Archetti, I. Scheuring, M. Hoffman, M. E. Frederickson, N. E. Pierce, and D. W. Yu, "Economic game theory for mutualism and cooperation: Cooperation amongst non-kin and mutualism," *Ecology Letters*, vol. 14, pp. 1300–1312, Dec. 2011.

[43] M. Begon, J. L. Harper, C. R. Townsend, *et al.*, *Ecology: individuals, populations and communities.* No. ED. 2, Blackwell Scientific Publications Ltd, 1990.

[44] M. L. Best, "Coevolving Mutualists Guide Simulated Evolution," in *Artificial Life VII* (M. A. Bedau, J. S. McCaskill, N. H. Packard, and S. Rasmussen, eds.), pp. 179–185, The MIT Press, Aug. 2000.

[45] S. Paracer and V. Ahmadjian, *Symbiosis: an introduction to biological associations*. Oxford University Press, USA, 2000.

[46] T. D. P. Brunet and W. F. Doolittle, "The generality of Constructive Neutral Evolution," *Biology & Philosophy*, vol. 33, p. 2, Apr. 2018.

[47] F. J. Weissing, P. Edelaar, and G. S. Van Doorn, "Adaptive speciation theory: a conceptual review," *Behavioral Ecology and Sociobiology*, vol. 65, pp. 461–480, Mar. 2011.

[48] S. J. Gould, *The panda's thumb: More reflections in natural history*. WW Norton & company, 2010.

[49] B. H. Burma and E. Mayr, "The Species Concept: A Discussion," *Evolution*, vol. 3, p. 369, Dec. 1949.

[50] U. Dieckmann and M. Doebeli, "On the origin of species by sympatric speciation," *Nature*, vol. 400, no. 6742, pp. 354–357, 1999.

[51] J. A. Coyne and H. A. Orr, "Speciation: a catalogue and critique of species concepts," *Philosophy of biology: an anthology*, pp. 272–92, 2004.

[52] R. M. Brucker and S. R. Bordenstein, "Speciation by symbiosis," *Trends in ecology & evolution*, vol. 27, no. 8, pp. 443–451, 2012.

[53] L. Margulis and D. Sagan, *Acquiring genomes: A theory of the origin of species*. Basic books, 2008.

[54] U. Kutschera and K. J. Niklas, "Endosymbiosis, cell evolution, and speciation," *Theory in Biosciences*, vol. 124, pp. 1–24, 2005.

[55] D. R. Finn, M. App, L. Hertzog, and C. C. Tebbe, "Reconciling concepts of black queen and tragedy of the commons in simulated bulk soil and rhizosphere prokaryote communities," *Frontiers in Microbiology*, vol. 13, p. 969784, Sept. 2022.

[56] A. W. Lo, "Reconciling Efficient Markets with Behavioral Finance: The Adaptive Markets Hypothesis," *Journal of investment consulting*, 2005.

[57] C. De Mazancourt and M. W. Schwartz, "A resource ratio theory of cooperation," *Ecology Letters*, vol. 13, pp. 349–359, Mar. 2010.

[58] L. Bull and T. C. Fogarty, "Artificial symbiogenesis," *Artificial Life*, vol. 2, no. 3, pp. 269–292, 1995.

[59] L. Margulis, "Symbiosis and evolution," *Scientific American*, vol. 225, no. 2, pp. 48–61, 1971.

[60] R. Guerrero, L. Margulis, M. Berlanga, *et al.*, "Symbiogenesis: the holobiont as a unit of evolution," *Int Microbiol*, vol. 16, no. 3, pp. 133–143, 2013.

[61] L. Bull, "Artificial Symbiogenesis and Differing Reproduction Rates," *Artificial Life*, vol. 16, pp. 65–72, Jan. 2010.

[62] C. T. Brown, L. A. Hug, B. C. Thomas, I. Sharon, C. J. Castelle, A. Singh, M. J. Wilkins, K. C. Wrighton, K. H. Williams, and J. F. Banfield, "Unusual biology across a group comprising more than 15% of domain Bacteria," *Nature*, vol. 523, pp. 208–211, July 2015.

[63] D. Fudenberg and J. Tirole, *Game theory*. MIT press, 1991.

[64] J. García and M. Van Veelen, "No strategy can win in the repeated prisoner's dilemma: linking game theory and computer simulations," *Frontiers in Robotics and AI*, vol. 5, p. 102, 2018.

[65] G. Hardin, "The tragedy of the commons: the population problem has no technical solution; it requires a fundamental extension in morality.," *science*, vol. 162, no. 3859, pp. 1243–1248, 1968.

[66] F. Dionisio and I. Gordo, "The tragedy of the commons, the public goods dilemma, and the meaning of rivalry and excludability in evolutionary biology," *Evolutionary Ecology Research*, vol. 8, no. 2, pp. 321–332, 2006.

[67] W. F. Lloyd, *Two lectures on the checks to population*. JH Parker, 1833.

[68] J. Apaloo, J. S. Brown, G. G. McNickle, T. L. Vincent, and T. L. Vincent, "Ess versus nash: solving evolutionary games," *Evolutionary Ecology Research*, vol. 16, no. 4, pp. 293–314, 2015.

[69] D. F. Toupo and S. H. Strogatz, "Nonlinear dynamics of the rock-paper-scissors game with mutations," *Physical Review E*, vol. 91, no. 5, p. 052907, 2015.

[70] R. Axelrod and W. D. Hamilton, "The evolution of cooperation," *science*, vol. 211, no. 4489, pp. 1390–1396, 1981.

[71] M. A. Nowak, "Five Rules for the Evolution of Cooperation," *Science*, vol. 314, pp. 1560–1563, Dec. 2006.

[72] J. L. Sachs, U. G. Mueller, T. P. Wilcox, and J. J. Bull, "The evolution of cooperation," *The Quarterly review of biology*, vol. 79, no. 2, pp. 135–160, 2004.

[73] B. Wilder and K. O. Stanley, "Altruists Proliferate Even at a Selective Disadvantage within Their Own Niche," *PLOS ONE*, vol. 10, p. e0128654, June 2015.

[74] D. C. Vural, A. Isakov, and L. Mahadevan, "The organization and control of an evolving interdependent population," *Journal of The Royal Society Interface*, vol. 12, p. 20150044, July 2015.

[75] M. Salahshour, "Evolution of cooperation in costly institutions exhibits Red Queen and Black Queen dynamics in heterogeneous public goods," *Communications Biology*, vol. 4, p. 1340, Nov. 2021.

[76] M. Perc and A. Szolnoki, "Coevolutionary games - a mini review," *Biosystems*, vol. 99, pp. 109–125, Feb. 2010. arXiv:0910.0826 [cond-mat, physics:nlin, physics:physics, q-bio].

[77] G. Chomicki, E. T. Kiers, and S. S. Renner, "The Evolution of Mutualistic Dependence," *Annual Review of Ecology, Evolution, and Systematics*, vol. 51, pp. 409–432, Nov. 2020.

[78] J. Lehman, J. Clune, D. Misevic, C. Adami, L. Altenberg, J. Beaulieu, P. J. Bentley, S. Bernard, G. Beslon, D. M. Bryson, N. Cheney, P. Chrabaszcz, A. Cully, S. Doncieux, F. C. Dyer, K. O. Ellefsen, R. Feldt, S. Fischer, S. Forrest, A. Fŕenoy, C. Gagńe, L. Le Goff, L. M. Grabowski, B. Hodjat, F. Hutter, L. Keller, C. Knibbe, P. Krcah, R. E. Lenski, H. Lipson, R. MacCurdy, C. Maestre, R. Miikkulainen, S. Mitri, D. E. Moriarty, J.-B. Mouret, A. Nguyen, C. Ofria, M. Parizeau, D. Parsons, R. T. Pennock, W. F. Punch, T. S. Ray, M. Schoenauer, E. Schulte, K. Sims, K. O. Stanley, F. Taddei, D. Tarapore, S. Thibault, R. Watson, W. Weimer, and J. Yosinski, "The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities," *Artificial Life*, vol. 26, pp. 274–306, May 2020.

[79] J. Skalse, N. Howe, D. Krasheninnikov, and D. Krueger, "Defining and characterizing reward gaming," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9460–9471, 2022.

[80] S. W. Mahfoud, "Niching Methods for Genetic Algorithms," 1995.

[81] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[82] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," *Artificial intelligence*, vol. 40, no. 1-3, pp. 235–282, 1989.

[83] W. Rafajłowicz, "Horizontal gene transfer as a method of increasing variability in genetic algorithms," in *Artificial Intelligence and Soft Computing: 17th International Conference, ICAISC 2018, Zakopane, Poland, June 3-7, 2018, Proceedings, Part I 17*, pp. 505–513, Springer, 2018.

[84] A. Telikani, A. Tahmassebi, W. Banzhaf, and A. H. Gandomi, "Evolutionary machine learning: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–35, 2021.

[85] M. Ester, H.-P. Kriegel, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," vol. 96, no. 34, pp. 226–231, 1996.

[86] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN," *ACM Transactions on Database Systems*, vol. 42, pp. 1–21, Sept. 2017.

[87] M. A. Potter and K. A. Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature — PPSN III* (G. Goos, J. Hartmanis, J. Leeuwen, Y. Davidor, H.-P. Schwefel, and R. Männer, eds.), vol. 866, pp. 249–257, Berlin, Heidelberg: Springer Berlin Heidelberg, 1994. Series Title: Lecture Notes in Computer Science.

[88] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, vol. 2, (Seoul, South Korea), pp. 1101–1108, IEEE, 2001.

[89] M. A. Potter and K. A. D. Jong, "Evolving neural networks with collaborative species," pp. 340–345, 1995.

[90] K. O. Stanley and R. Miikkulainen, "Evolving Neural Networks through Augmenting Topologies," *Evolutionary Computation*, vol. 10, pp. 99–127, June 2002.

[91] K. Stanley and R. Miikkulainen, "Efficient evolution of neural network topologies," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, vol. 2, (Honolulu, HI, USA), pp. 1757–1762, IEEE, 2002.

[92] N. Garcia-Pedrajas, C. Hervas-Martinez, and J. Munoz-Perez, "COVNET: a cooperative coevolutionary model for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 14, pp. 575–596, May 2003.

[93] M. Gong, J. Liu, A. K. Qin, K. Zhao, and K. C. Tan, "Evolving Deep Neural Networks via Cooperative Coevolution With Backpropagation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 420–434, Jan. 2021.

[94] D. E. Moriarty and R. Mikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Machine learning*, vol. 22, no. 1, pp. 11–32, 1996.

[95] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks," *Artificial Life*, vol. 15, pp. 185–212, Apr. 2009.

[96] F. vandenBergh and A. Engelbrecht, "A Cooperative Approach to Particle Swarm Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 225–239, June 2004.

[97] M. A. Meselhi, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Contribution Based Co-Evolutionary Algorithm for Large-Scale Optimization Problems," *IEEE Access*, vol. 8, pp. 203369–203381, 2020.

[98] T. Eguchi, K. Hirasawa, and Jinglu Hu, "Symbiotic evolutional models in multiagent systems," in *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, vol. 2, (Canberra, Australia), pp. 739–746, IEEE, 2003.

[99] T. Eguchi, K. Hirasawa, J. Hu, and N. Ota, "A study of evolutionary multiagent models based on symbiosis," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 36, pp. 179–193, Feb. 2006.

[100] J. Shao, Z. Lou, H. Zhang, Y. Jiang, S. He, and X. Ji, "Self-Organized Group for Cooperative Multi-agent Reinforcement Learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5711–5723, 2022.

[101] C. H. Yong and R. Miikkulainen, "Cooperative Coevolution of Multi-Agent Systems," *University of Texas at Austin, Austin, TX*, 2001.

[102] R. Dreżewski, "A Model of Co-evolution in Multi-agent System," in *Multi-Agent Systems and Applications III* (V. Mařík, M. Pěchouček, and J. Müller, eds.), vol. 2691, pp. 314–323, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. Series Title: Lecture Notes in Computer Science.

[103] J. Y. Kim and Y. K. Kim, "Multileveled Symbiotic Evolutionary Algorithm: Application to FMS Loading Problems," *Applied Intelligence*, vol. 22, pp. 233–249, May 2005.

[104] D. Klijn and A. E. Eiben, "A coevolutionary approach to deep multi-agent reinforcement learning," Apr. 2021. arXiv:2104.05610 [cs].

[105] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Qin, "Benchmark Functions for the CEC'2013 Special Session and Competition on Large-Scale Global Optimization," *gene*, vol. 7, no. 33, p. 8, 2013.

[106] M. Helbig and A. Engelbrecht, "Benchmark functions for cec 2015 special session and competition on dynamic multi-objective optimization," *Dept. Comput. Sci., Univ. Pretoria, Pretoria, South Africa, Rep*, 2015.

[107] X. Pan, M. Liu, F. Zhong, Y. Yang, S.-C. Zhu, and Y. Wang, "MATE: Benchmarking Multi-Agent Reinforcement Learning in Distributed Target Coverage Control," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27862–27879, 2022.

[108] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks," *arXiv preprint arXiv:2006.07869*, 2020.

[109] T. R. Balch, *Behavioral diversity in learning robot teams*. Georgia Institute of Technology, 1998.

[110] M. A. Potter, L. A. Meeden, A. C. Schultz, *et al.*, "Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists," in *International joint conference on artificial intelligence*, vol. 17, pp. 1337–1343, Citeseer, 2001.

[111] M. Waibel, L. Keller, and D. Floreano, "Genetic Team Composition and Level of Selection in the Evolution of Cooperation," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 648–660, June 2009.

[112] R. Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms," *Biosystems*, vol. 39, pp. 263–278, Jan. 1996.

[113] D. Gupta and S. Ghafir, "An overview of methods maintaining diversity in genetic algorithms," *International journal of emerging technology and advanced engineering*, vol. 2, no. 5, pp. 56–60, 2012.

[114] S. Wright *et al.*, "The roles of mutation, inbreeding, crossbreeding, and selection in evolution," 1932.

[115] P. B. Grosso, *Computer simulations of genetic adaptation: Parallel subcomponent interaction in a multilocus model*. University of Michigan, 1985.

[116] A. Hara, "Emergence of cooperative behavior using adg; automatically defined groups," in *GECCO-99: Proc. Genetic and Evolutionary Computation Conference*, pp. 1039–1046, Morgan Kaufmann, 1999.

[117] J. C. Bongard, "The legion system: A novel approach to evolving heterogeneity for collective problem solving," in *European Conference on Genetic Programming*, pp. 16–28, Springer, 2000.

[118] J. C. Bongard, "Reducing Collective Behavioural Complexity through Heterogeneity," in *Artificial Life VII* (M. A. Bedau, J. S. McCaskill, N. H. Packard, and S. Rasmussen, eds.), pp. 327–336, The MIT Press, Aug. 2000.

[119] G. Nitschke, M. Schut, and A. Eiben, "Evolving behavioral specialization in robot teams to solve a collective construction task," *Swarm and Evolutionary Computation*, vol. 2, pp. 25–38, Feb. 2012.

[120] J. Gomes, P. Mariano, and A. L. Christensen, "Dynamic Team Heterogeneity in Cooperative Coevolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 22, pp. 934–948, Dec. 2018.

[121] D. H. Wolpert and K. Tumer, "Optimal payoff functions for members of collectives," *Advances in Complex Systems*, vol. 4, no. 02n03, pp. 265–279, 2001.

[122] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[123] C. Nordahl, V. Boeva, H. Grahn, and M. Persson Netz, "EvolveCluster: an evolutionary clustering algorithm for streaming data," *Evolving Systems*, vol. 13, pp. 603–623, Aug. 2022.

[124] T. Anwar, S. Nepal, C. Paris, J. Yang, J. Wu, and Q. Z. Sheng, "Tracking the Evolution of Clusters in Social Media Streams," *IEEE Transactions on Big Data*, vol. 9, pp. 701–715, Apr. 2023.

[125] D. Barbara and P. Chen, "Tracking Clusters in Evolving Data Sets," pp. 239–243, 2001.

[126] T. Haynes and S. Sen, "Crossover Operators for Evolving A Team," *Genetic programming*, vol. 199, 1997.

[127] J. Gomes, P. Mariano, and A. L. Christensen, "Novelty-Driven Cooperative Coevolution," *Evolutionary Computation*, vol. 25, pp. 275–307, June 2017.

[128] S. Luke and L. Spector, "Evolving teamwork and coordination with genetic programming," *Genetic Programming*, vol. 96, pp. 150–56, 1996.

[129] F. Ritz, D. Ratke, T. Phan, L. Belzner, and C. Linnhoff-Popien, "A Sustainable Ecosystem through Emergent Cooperation in Multi-Agent Reinforcement Learning," vol. 2021, no. 1, p. 74, 2021.

[130] J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, eds., *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. The MIT Press, 1996.

[131] R. Braakman, M. J. Follows, and S. W. Chisholm, "Metabolic evolution and the self-organization of ecosystems," *Proceedings of the National Academy of Sciences*, vol. 114, Apr. 2017.

[132] S. G. Ficici and J. B. Pollack, "A Game-Theoretic Approach to the Simple Coevolutionary Algorithm," in *Parallel Problem Solving from Nature PPSN VI* (G. Goos, J. Hartmanis, J. Van Leeuwen, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, eds.), vol. 1917, pp. 467–476, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. Series Title: Lecture Notes in Computer Science.

[133] H. S. Ramadan, H. A. Maghawry, M. El-Eleamy, and K. El-Bahnasy, "A HEURISTIC NOVEL APPROACH FOR DETERMINATION OF OPTIMAL EPSILON FOR DBSCAN CLUSTERING ALGORITHM," . *Vol.*, no. 7, 2022.

[134] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, pp. 1492–1496, June 2014.

[135] A. Pocheville, "The ecological niche: history and recent controversies," *Handbook of evolutionary thinking in the sciences*, pp. 547–586, 2015.

Appendices

A Glossary — Terms List

Throughout this thesis a lot of terminology is used. The problem with terminology is that the meaning is not always clear, different words can be used for the same thing, and one word can be used to mean different things. In an attempt to improve clarity to the reader we attempted to stay consistent with the uses of terms, and provide an overview of most important terms.

Term	Meaning
Emergent Symbiotic Speciation	The process through which new species evolve from a shared environment and develop a mutualistic relationship.
Hologenome	The collective genome of a host and its symbiotic microbes, considered together as a unit of selection in evolutionary processes.
Cooperative Coevolution	An evolutionary computation strategy where complex problems are decomposed and solved through the combination of coevolved species / subcomponents.
Functional Heterogeneity	Diversity of functional roles within a system.
Evolutionary Pathway	The conceptual path of increasing fitness on the fitness landscape from one point on the sequence space to another.
Holobiont	A host organism and its symbiotic species as discrete ecological unit.
Composite Organism / Entity	A collection of symbiotic species as descrete ecological unit, without host-symbiont hierarchical relationship.
Problem Decomposition	Breaking up a problem into a collection of sub-problems, that if individually solved solves the complete problem.
Parapatric Speciation	Speciation through population division and reproductive isolation, but with a level of gene exchange.
Allopatric Speciation	Speciation through population division and total reproductive isolation.
Sympatric Speciation	Speciation without reproductive isolation, e.g., through competition over limited resources or function loss.
Credit Assignment	Determining the value of one individuals contribution to the collective result.
Genome Distance-Based Mating	An assortative mating strategy that selects partners based on genetic distance.
Species Representation Fitness Scaling	Fitness adjustment strategy based on the proportional difference in species representation in the individual- and composition genome population.
Evolutionary Compositions	Genetic algorithm based agent type sampling and composition forming strategy.

Term	Meaning
Fitness Landscape	A conceptual landscape representing how different genotypes correspond to their reproductive success or fitness.
Niche	The match of a species to the environment[135]. A local optimum / peak region on the fitness landscape.
Local Optimum	The position of a relative extremum (maximum or minimum) value of a function. The global optimum is the extremum of all possible values, local is the extremum of its neighbourhood.
Punctuated Equilibrium	Theory of species evolution through long periods of evolutionary stability interspersed with short bursts of rapid evolutionary change. Contra evolutionary gradualism.
Quasispecies	A large group of related genotypes with high rate of change within and between generations.
Symbiosis	Two or more species living in close association.
Mutualism	A mutually beneficial symbiotic relationship. If the mutualistic relationship is obligate, species are entirely dependent on each other for survival. If it is facultative, they are not.
Commensalism	A symbiotic relationship beneficial to one side and neutral to the other.
Cladogenesis	The splitting of one species into two distinct species.
Symbiogenesis	The process by which symbiotic partners combine to a single organism.[61]
Pay-off Matrix	An abstract representation of interaction outcomes as payoff between two or more players.
Maximal Social Welfare	In game theory, the point on the pay-off matrix giving the largest collective reward.
Pareto Optimal / Efficient	The situation where there is no other option where all players do at least as well and at least one player has a higher payoff.
Nash Equilibrium	The situation where no player gains by changing their strategy as long as the other player keeps their strategy unchanged.
Satisfying Function	A function where an increase in input does not yield a change in the output past the point of satisfaction.
Reward Hacking	Optimizing the literal definition of an objective function without fulfilling the intended goal.
Agent	An individual autonomous entity (inter)acting based on input from the environment, including other agents.
Separability	The degree to which a thing can be decomposed into independent subcomponents.

Term	Meaning
Epistasis / Linkage / Gene interaction	Interaction between genes or variables. The dependent relationship between subcomponents.
(En)Forced Subpopulations	Evolving multiple species in separate gene populations.
Island Model	A model of forced subpopulations with limited population crossover.
Individual Fitness	The direct representation of the performance of an individual, the local reward.
Collective Fitness	The representation of the performance of all contributing individuals, the global reward.
Personal Fitness	The final fitness attributed to an individual that is used in selection, which can be a combination of individual and collective fitness.
Unsupervised Clustering	The process of identifying groups of similar data points in a datasets in bottom-up fashion.

Table 5: Terms and meanings.

B Model Parameters and Configurations

An overview of the parameters and model configurations is provided in the following tables, along with their effect and default values. Values often deviate from the default settings across experiments for various purposes. Where applicable, deviations are mentioned.

Parameter Name	Description	P	T	F
n runs	Number of complete model runs.	1	10	10
n generations	Number of generations in a model run.	500	500	500
n chromosomes	Agent chromosome population size.	150	300	300
n compositions	Number of compositions to run per generation. Composition chromosome population size.	100	30	30
runs per composition	Number of times each composition is run in the domain model.	5	1	1

Table 6: General model settings. P = Predator-Prey, T = Toxin, F = Function Optimization.

Parameter Name	Type	Options / Range	Description	(indirect) Effects	Default Value
gene data type	nominal	continuous	Gene values are continuous floating points		model dependent
		binary	Gene values are binary (0, 1)		
		nominal	Gene values are nominal		
reproduce fraction	closed unit interval	[0,1]	Fraction of the top fitness individuals that are selected for reproduction	Truncation selection	0.5
cull fraction	closed unit interval	[0,1]	Fraction of the bottom fitness individuals that are removed from the population. The inverse fraction remains (elitism).	Elitist selection	model dependent
selection type	nominal	random	First parents are randomly selected from the reproducing individuals		fitness proportional
		fitness proportional	First parents are selected randomly weighted with their fitness from the reproducing individuals		
mate selection type	nominal	random	Second parents are selected randomly from the set of reproducing individuals		experiment dependent
		fitness proportional	Second parents are selected randomly weighted with their fitness from the reproducing individuals		
		approximate nearest neighbour	Second parents are selected according to the smallest euclidian distance to the first parent from sample.		
mate selection sample size	continuous	[0,N]	Size of sample for approximate nearest neighbour mate selection.	Scale between random and certain nearest neighbour.	50, population size dependent
allow self mating	binary	{0,1}	Allow first and second parents to be the same chromosome.	Allow mutation only variation.	false
mutation probability	closed unit interval	[0,1]	Probability for a gene to mutate in newly produced offspring. Applies to each gene individually.	Exploration / exploitation tradeoff	1 / chromosome length
mutation range	continuous	$\mathbb{Q}$	Range of the mutation size. Mutation size sampled from uniform distribution from [-range, +range].	Search space step size.	0.1
knockout mutation	binary	{0,1}	Allow knockout mutations to occur, disabling a gene expression. Reverseable action.		model dependent
knockout mutation probability	closed unit interval	[0,1]	Probability for a gene knockout mutation to occur in newly produced offspring. Applies to each gene individually.		1 / chromosome length
initial gene knockout	continuous	$\mathbb{W}$	Number of genes knocked out for chromosomes in the initial population. 0 gives a random number on the range [0, chromosome length]. Only applies if knockout mutations are enabled		0
chromosome resizing	binary	{0,1}	Allows gene deletion and insertion. (used for compositions)		model dependent
chromosome resizing probability	closed unit interval	[0,1]	Probability for a gene to be deleted or inserted on a chromosome. If the action occurs, there is equal probability for either deletion or insertion. Probability applies to chromosome.		0.1

Table 7: Agent Evolution Parameters.

Parameter Name	Options / Range	Description	Default Value
gene data type	nominal	Gene values are nominal values	nominal
reproduce fraction	[0,1]	Fraction of the top fitness individuals that are selected for reproduction	0.5
cull fraction	[0,1]	Fraction of the bottom fitness individuals that are removed from the population. The inverse fraction remains (elitism).	1.0
selection type	random	First parents are randomly selected from the reproducing individuals	fitness proportional
	fitness proportional	First parents are selected randomly weighted with their fitness from the reproducing individuals	
mate selection type	random	Second parents are selected randomly from the set of reproducing individuals	fitness proportional
	fitness proportional	Second parents are selected randomly weighted with their fitness from the reproducing individuals	
allow self mating	{0,1}	Allow first and second parents to be the same chromosome.	true
mutation probability	[0,1]	Probability for a gene to mutate in newly produced offspring. Applies to each gene individually.	1 / chromosome length
mutation range	$\mathbb{Q}$	Range of the mutation size. Mutation size sampled from uniform distribution from [-range, +range].	0.1
chromosome resizing	{0,1}	Allows gene deletion and insertion. (used for compositions)	true
chromosome resizing probability	[0,1]	Probability for a gene to be deleted or inserted on a chromosome. If the action occurs, there is equal probability for either deletion or insertion. Probability applies to chromosome.	0.1

Table 8: Composition Evolution Parameters.

C Code

The complete source code of the project can be found on GitHub:

https://github.com/IvoSte/EmergentSymbioticSpeciation

Here, the exact setup and results of a large amount of experiments can be found, including the ones presented in the results section. A read-me explaining how to run experiments is available.

Parameter Name	Options / Range	Description	Default value
population type	forced subpopulations	Breed populations separately with fixed species assignment per subpopulations	experiment dependent
	emergent subpopulations	Breed populations from a single gene pool and detect emerging species	
n subpopulations	$\mathbb{N}$	The number of separate chromosome populations	
chromosome sampling	simple random	Sample individuals to fill domain model slots randomly with replacement after exhaustion	population type dependent
	deterministic	Sample individuals using a deterministic algorithm, see section 4.3.6	
	evolutionary compositions	Sample individuals using evolutionary compositions, see section 4.3.7	
composition fitness scaling	{0,1}	Apply a species representation scalar to fitness of individuals, see section 4.3.8	experiment dependent
species detection method	DBSCAN	DBSCAN is used as vector quantization method to assign a species label to clusters of chromosomes.	chromosome type dependent
	binary	The binary encoding of a chromosome is the species	
	binary (knockout)	The binary encoding of all non-knocked out genes of a chromosome is the species	
species detection ε (DBSCAN)	$\mathbb{Q}$	The maximum distance between two points to be considered a neighbourhood	0.5
species detection minimum samples	$\mathbb{N}$	The minimum number of samples for a neighbourhood to be considered a cluster	1
species tracking	{0,1}	Remember previously identified species and use this information to assign species labels to clusters.	chromosome type dependent
species tracking search depth	$\mathbb{N}$	Number of generations back to compare new prototypes to.	5

Table 9: Speciation model settings. Note RC experiments run emergent subpopulations with simple random chromosome sampling.

Parameter Name	Options / Range	Description	Default value
n agents	$\mathbb{N}$	Number of predators	3
n prey	$\mathbb{N}$	Number of prey	1
prey behaviour	flee from closest predator	Prey moves opposite closest predator	flee from closest predator with noise
	flee from closest predator with noise	Prey moves opposite closest predator with heading noise	
individual fitness type	individual kills	The number of catches by this agent	catch with distance
	catch with distance	The fitness calculation eq. (9)	
collective fitness type	collective kills	The number of catches by all agents	catch with distance
	catch with distance	The fitness calculation eq. (9), averaged for all agents, eq. (10)	
collective fitness weight	[0,1]	The scalar determining the individual and collective fitness weight.	0

Table 10: Predator-Prey Domain Model parameters.

Parameter Name	Options / Range	Description	Default value
n agents ($\|A\|$)	$\mathbb{N}$	Number of agents that can have the function to clean toxin from the environment.	30
n toxins ($\|T\|$)	$\mathbb{N}$	Number of toxins in the environment.	5
toxin base (c_B)	$\mathbb{N}$	Base toxin level amount	10
toxin cleanup rate (R)	$\mathbb{N}$	Amount of toxin removed from the environment by a single agent with the cleanup function for that toxin.	1
function cost (C_f)	$\mathbb{N}$	The fitness cost for having the function to clean toxin from the environment.	3
function cost multiplier (M)	$\mathbb{Q}$	Cost increase or decrease exponent for each subsequent active function.	1.0
individual fitness type	default	The combined function cost and damage from remaining toxins, see eq. (15)	'default'
collective fitness type	average fitness	The average individual fitness of all agents, see eq. (16)	toxin remainder
	toxin remainder	The sum of all remaining toxin in the system, see eq. (17)	
collective fitness weight	[0,1]	The scalar determining the individual and collective fitness weight.	0

Table 11: Toxin Domain Model parameters.

Parameter Name	Options / Range	Description	Default value
function	see eqs. (20) to (23)	The optimization function used	experiment dependent
n dimensions (N)	$\mathbb{N}$	The number of dimensions, the length of the decision vector.	10
n agents	$\mathbb{N}$	The number of subcomponents in a full solution. Variable if composition resizing is enabled.	10
n genes	$\mathbb{N}$	The number of genes on a subcomponent chromosome, equal to the number of decision variables of the optimization function.	10
function cost	$\mathbb{N}$	Fitness penalty for each non-knocked out gene.	0
individual fitness type	default	The signed objective value minus total function cost, see eq. (31)	'default'
collective fitness type	average	The signed objective value, see eq. (32)	objective value
collective fitness weight	[0,1]	The scalar determining the individual and collective fitness weight.	0

Table 12: Function Optimization Domain Model parameters.