Lennart Hoffmann

# Multi-objective optimisation for evaluation of collective machine consciousness

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

Intelligent Cooperative Systems
Intelligent Systems Group

# Multi-objective optimisation for evaluation of collective machine consciousness

Master's Thesis

Lennart Hoffmann

2018-11-08

Professor: Prof. Dr. Sanaz Mostaghim

**Lennart Hoffmann:** *Multi-objective optimisation for evaluation of collective machine consciousness*
Otto-von-Guericke Universität
Intelligent Cooperative Systems
Intelligent Systems Group
Magdeburg, 2018/2019.

# Abstract

Artificial Intelligence (AI) is becoming more and more complex every day, while simultaneously becoming more important for our everyday life. A day without interacting with an AI has become a rarity for most people. But what if an AI we're engaging with is not just a lifeless form of machinery, but a conscious being, with thoughts and feelings of its own? The resulting ethical problems of this scenario could have enormous consequences; nations could create new laws limiting and regulating the use of AI, which would change a lot of things we became accustomed to over the last few years. Before any of that happens, one would first have to establish whether a machine is conscious or not. However, there are not a lot of methods for testing the consciousness of a machine at the moment of writing this thesis. One of the most promising approach is the Integrated Information Theory (IIT) 3.0 developed by Professor Tononi and his students [61], which aims to make consciousness testable.

Using the integrated information theory, the goal of this thesis is to investigate the consciousness, measured by their integrated information, of a swarm of evolved organisms interacting with each other over the course of a multi-objective evolutionary algorithm. The basis for that is a simulation based on Fischer [32], in which organisms are placed in one of two rooms, which are connected through a gate passage, but are otherwise surrounded by an impervious wall. The organisms have three different objectives: Maximise goal passages, minimise collisions and minimise movement penalties. To gain a better understanding of the integrated information of the evolved organisms, two different selection algorithms are used throughout this thesis. The first selection algorithm is based on the nondominated sorting genetic algorithm II (NSGA-II) by Deb et al. [25], while the second is a decentralised algorithm operating on a solution's neighbourhood, which was developed for this thesis. Comparing the results of these two fundamentally different approaches should allow for insights about what influence the choice of selection algorithm has on the end result, thus assisting in understanding how the consciousness of a solution evolves over time.

# Contents

# List of Figures

# List of Tables

# Table of Acronyms

| | |
|---|---|
| **IIT** | Integrated Information Theory |
| **MICS** | Maximally Irreducible Conceptual Structure |
| **MICE** | Maximally Irreducible Cause and Effect |
| **ci** | cause information |
| **ei** | effect information |
| **cei** | cause-effect information |
| **EMD** | Earth Mover's Distance |
| **MIP** | Minimum Information Partition |
| **CI** | Conceptual Information |
| **MB** | Markov Brain |
| **MC** | Main Complex |
| **ANN** | Artificial Neural Network |
| **MABE** | Modular Agent Based Evolver |
| **TPM** | Transition Probability Matrix |
| **CM** | Connectivity Matrix |
| **SEM** | Standard Error of the Mean |
| **HV** | Hypervolume |

| | |
|---|---|
| **GD** | Generational Distance |
| **NSGA** | nondominated sorting genetic algorithm |
| **SPEA** | Strength Pareto Evolutionary Algorithm |
| **MOEA/D** | Multi-objective evolutionary algorithm based on decomposition |
| **MOGLS** | Multi-objective genetic local search algorithm |
| **NSGA-II** | nondominated sorting genetic algorithm II |
| **AI** | Artificial Intelligence |
| **EA** | Evolutionary Algorithm |
| **GA** | Genetic Algorithm |
| **MOEA** | Multi-objective Evolutionary Algorithm |
| **HSO** | Hypervolume by Slicing Objectives |
| **SSEA** | Spatially structured evolutionary algorithm |
| **MOCell** | Multi-objective cellular genetic algorithm |
| **TSP** | Travelling salesman problem |

# 1 Introduction

This chapter will act as an introduction to this thesis. After motivating the research of machine consciousness and other work done in this thesis, the goals of this work will be explained. Lastly, the structure of this thesis will be presented.

## 1.1 Motivation

One of the research areas currently receiving the most attention by both the scientific world and the media is AI. A lot of hugely successful companies, like Google or Amazon, are investing a substantial amount of money and time into the research of AI products. But AI can't just be found in these newly developed items, it is also slowly being integrated in some of the more common-place products like refrigerators or dish washers [30]. In addition to AI becoming close to ever-present in the life of many people, it is also getting more complex every day.

At the time of writing this thesis, most AI found in the previously mentioned products can easily be identified as artificial, but given the current rate of development, the rise in complexity could one day lead to bigger and bigger difficulties differentiating human and machine. After reaching a certain complexity, one has to ask whether the AI we're interacting with every day might actually have become conscious. If it was, a lot of discussions about machine ethics would have to be held, which would eventually lead to huge changes to the life we became used to in the last few years. However, one would first have to prove with certainty that an Artificial Intelligence has given rise to consciousness.

There are a lot of theories about the consciousness of machines [72][86][12]. Without an easy way to create conscious individuals, testing them is very hard though. One would have to rely on experiments on live humans or other conscious organisms, which limits the research possibilities and increases the time, cost and effort needed for testing. Simulating a population of conscious individuals or even just one conscious organism, which would both also be useful for a lot of studies in the field of neuroscience, is a problem that doesn't seem to have a good universal solution yet. With the help of such a simulation, the internal structures of an individual or the environment the population is simulated in could easily be changed, which could make testing these theories a lot easier.

One theory that aims to make consciousness testable, which would thus help with the aforementioned problems, is the Integrated Information Theory 3.0 by Tononi et al [61]. Taking the opposite approach to other neuroscientific theories about consciousness, IIT starts from a set of axioms describing the fundamental characteristics of consciousness and translates these into postulates, which outline the conditions that a physical mechanism, e.g. a neuron and its connections, has to fulfil to be able to give rise to consciousness. According to IIT, an experience is defined in its quality by its *Maximally Irreducible Conceptual Structure (MICS)*, while the experience's intensity, or quantity, is specified through its integrated information $\Phi^{Max}$.

There are existing works [44][29], which show that the integrated information of a simulated adaptive system increases with its fitness over the course of an evolutionary algorithm. However, the organisms, or animats, in these works were simulated individually and thus only interact with their environment and not with each other. In Fischer [32] this topic was investigated further by observing how the system's integrated information evolves within a group of animats that can interact with each other. This interaction could only occur between copies of the organism though.

## 1.2 Goals of this work

Collective behaviour between animals can form in various different ways. Some examples are species of birds travelling in a group during foraging or ants building nests. Depending on the species, this cooperation can have multiple advantages such as being able spot and fight predators more easily [55] or an increased efficiency in finding food [65]. However, collective behaviour can also occur between animals of different species, e.g. between clownfish and sea anemone. As a group they exhibit a better fitness and can be considered more intelligent than alone. In this work, we want to explore how the integrated information of such a collaborative, heterogeneous set of organisms evolves when presented with different environmental conditions.

The main goal of this thesis is thus to examine the development of the integrated information of a heterogeneous group of organisms interacting with each other over the course of an evolutionary algorithm. A simulation was developed as an extension of Fischer's work [32]. Instead of confining organisms to one isolated simulation per organism, the whole population will be placed inside one simulation, allowing for interactions between any organism. To keep the diversity of the population high, a multi-objective optimisation strategy will be used in this work. For this, three different, conflicting objectives were identified. This should result in a more heterogeneous set of organisms, which could help alleviate problems like premature convergence due to organisms being stuck in a local minimum. Additionally, a higher diversity should also improve results of the crossover operator and makes it easier to cover more of the search space [79].

Various different simulation parameter combinations were tested to determine the impact of each of them on the resulting performance and integrated information. One of these parameters controls the selection algorithm used in the evaluation phase of the evolutionary process, for which two fundamentally different approaches were developed. The first algorithm considers and modifies a globally accessible pool of organisms, while the second divides the population into neighbourhoods and applies the genetic operators on a more local scale. By comparing the results of these two contrasting methods and examining the influence of the choice of selection algorithm, we hope to gain a better understanding of how the integrated information of such a population evolves over time.

## 1.3 Structure of this thesis

This thesis is structured as follows:

**Chapter 2** gives an overview over the themes and the related work of this thesis. First, some concepts of multi-objective optimisation will be explained, followed by a discussion of different performance measurements for the results of said optimisations. Based on this, the next section focuses on how these concepts are applied to Evolutionary Algorithms (EAs). After presenting the basics of EAs, two different, i.e. centralised and decentralised, kinds of multi-objective EAs will be described and discussed. Lastly, an overview over the concepts and applications of the integrated information theory 3.0 is given.

**Chapter 3** outlines the structure of the experiment. The chapter starts with a description of the design of the simulation, ranging from its spatial layout and the organisms inhabiting that world to the identified objectives and the selection algorithm developed for this thesis. Afterwards, the chapter goes into more detail about the data analysis of the simulation and the implementation of both.

**Chapter 4** contains the evaluation of the experiment presented in chapter 3 and is split into two parts. The first part focuses on the direct analysis of the multi-objective simulation by evaluating the distribution and convergence of the fronts of each simulation run. This directly leads into the second part, in which the connections between experimental settings and the resulting integrated information and the brain structure of some of the best performing organisms are analysed.

**Chapter 5** will sum up the results of this thesis and **Chapter 6** will provide an outlook on some of the possible future works.

# 2 Background

This chapter provides an overview over the concepts used in this thesis. After an introduction to the basics of multi-objective optimisation, we will explore various ways to evaluate the results of such an optimisation. This is followed by a discussion about EAs and how they can be used to generate solutions for multi-objective problems. Finally, we will describe the principles of the current iteration of IIT (3.0), which is followed by a brief discussion thereof.

## 2.1 Multi-objective Optimisation

Optimisation is the study of finding an optimal solution by either minimising or maximising at least one objective while simultaneously satisfying all given constraints [24]. To evaluate a solution for a given problem, a so-called objective function is used, which usually has one or more input values and a single output value. Finding the optimal solution thus means optimizing the objective function. While single-objective optimisation is concerned with optimising the value of just one objective, multi-objective optimisation involves a set of objectives which are subject to either minimisation or maximisation. Mathematically, a multi-objective optimisation problem can be defined as

$$min(\vec{f}(\vec{x})) \qquad where: \vec{f}(\vec{x}) = [f_1(x), ..., f_m(x)], \tag{2.1}$$

where $m$ is the amount of objectives. Note that this definition assumes that every objective is subject to minimisation. An example of a multi-objective optimisation problem involving two objectives would be minimising cost while also maximising computing power when buying a new computer. However, in most applications the amount of objectives is way higher. Since the objective functions are often in conflict with each other, there is no single optimal solution. Optimising one objective usually entails a drop-off in quality in a different objective. Therefore, the general goal of multi-objective optimisation is to generate a diverse set of good solutions, from which an external decision maker, e.g. a human, can choose the final solution.

The population's "best" solutions can be found by applying the dominance criterion. When considering an optimisation problem whose objectives are all subject to minimisation as in 2.1, a solution $x_1$ dominates solution $x_2$

$$\begin{aligned} \text{if} \quad & \forall i \in \{1, ..., m\} : f_i(x_1) \leq f_i(x_2) \\ \text{and} \quad & \exists j \in \{1, ..., m\} : f_j(x_1) < f_j(x_2) \end{aligned} \tag{2.2}$$

In other words, a solution $x_1$ dominates solution $x_2$ if all of $x_1$'s objective values are at least equal to $x_2$'s with at least one value being better. A solution $x_1$ is called *non-dominated* or *Pareto optimal* if there is no other solution in the solution set $X$ that dominates $x_1$. The set of all Pareto optimal solutions is called the Pareto front [24]. An example of a solution set, in which one objective is maximised and one is minimised, and its corresponding Pareto front is shown in Figure 2.1.

### 2.1.1 Evaluation Methods

Two desirable qualities of a front are **diversity** and its **convergence** to the true Pareto front [50]. Diversity describes how spread out the solutions are across the different dimensions. The higher the distance between objective values, the higher, and thus better, the diversity. A high diversity allows the algorithm to explore more of the solution space, which could also help overcome local minima during the optimisation process. A front's convergence is basically defined by how close the solutions are to the true Pareto front, i.e. the optimal set of solutions. Both criteria are important for good results. High diversity alone doesn't help much if the front's values are far away from the optimum. Similarly, high convergence without the accompanying high diversity results in the solutions being extremely close together, thus making the final choice of the external decision maker close to irrelevant.

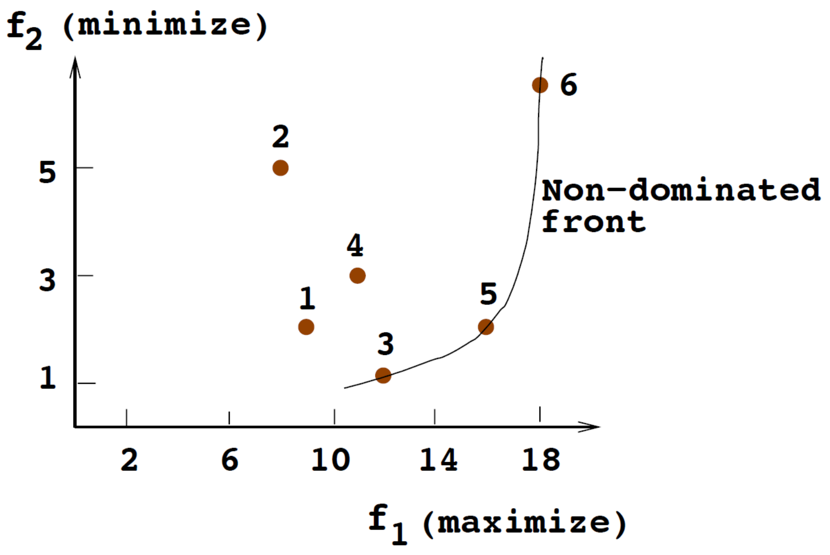


Figure 2.1: The solution set of a multi-objective optimisation problem with two objectives [88] plotted on a 2D scatter plot. The black line indicates the Pareto front of the set, i.e. the solutions that are not dominated by any other point in the set.

### Front spread and uniformity

The convergence of a front can be measured in multiple ways. One measure presented by Goel and Stander in their 2009 paper [37] is comparing how the number of non-dominated solutions evolves over the generations. The basic idea is that a smaller number of dominated solutions also implies a higher convergence. While simple, this measure can give the user an effective, computationally cheap overview over the evolutionary process. However, it doesn't necessarily show how the quality of the solutions evolves, since a solution set without any dominated organisms can still be far away from the true Pareto front.

Goel and Stander also introduce two different diversity metrics: The *front spread* and the *front uniformity*, which work in tandem with each other. The front spread is defined as the diagonal of the largest hypercube in the function space that includes all points. The longer the diagonal, the larger the spread of solutions and thus the diversity. A few extreme points can skew the results though, since such a point can increase the diagonal's length immensely, even when most of the solutions are close together. In such cases, the user can consult the uniformity measure. The uniformity measure estimates the homogeneity of the points in the Pareto optimal set. It is defined as:

$$\Delta = \sum_{i=1}^{N} \frac{|d_i - \bar{d}|}{N} \quad \text{where} \quad \bar{d} = \frac{1}{N} \sum_{i=1}^{N} d_i, \tag{2.3}$$

where $d_i$ is the crowding distance of an organism $i$. It behaves similarly to the standard deviation, i.e. a small uniformity is desired for a nicely distributed set of points. It also helps alleviate the extreme point issues, when combining it with the spread metric.

### Generational Distance

The Generational Distance (GD) [87] is a metric for measuring the distance from the current Pareto front $PF_{known}$ to the true Pareto front $PF_{true}$. As such it only measures the quality, i.e. convergence, of the solutions and not their diversity. It is one of the most commonly used Multi-objective Evolutionary Algorithm (MOEA) measures [68]. Mathematically, it can be defined as:

$$G = \frac{(\sum_{i=1}^{n} d_i^p)^{\frac{1}{p}}}{n}, \tag{2.4}$$

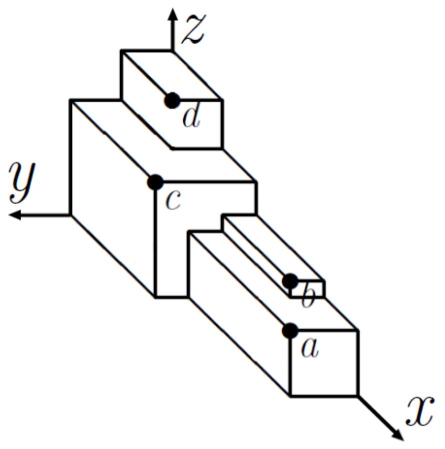
where $n$ is the size of generated Pareto front $PF_{known}$ and $d_i^p$ is the euclidean distance between point $p_i$ and the nearest point of $PF_{true}$. If $P_{true}$ is not known, one has to generate a set of reference points. For instance, the so-called 7-point [71] method achieves an approximation of the GD by generating equidistantly spaced points along each objective between the minimum and maximum possible values thereof. For a two-dimensional optimisation problem the seven points would consist of the origin point and three points for each of the objectives, i.e. one point at the objectives' maximum and the two evenly spaced points along the axis.

### Hypervolume

Another one of the most important and also widely used [68] convergence and diversity metrics used in multi-objective optimisation is the hypervolume [97].

Introduced in 1998 by Zitzlar and Thiele, it is also used as a selection criterion in some evolutionary algorithms [45][31] as a way apply to selection pressure, which rewards both convergence and solution diversity.

The hypervolume indicator represents the n-dimensional space covered by a n-dimensional set of points, i.e. a Pareto front of the population mapped to points in the objective space, relative to a chosen reference point. A popular method for choosing that reference point is taking the worst possible point in the objective space with respect to the objective functions and other constraints and shifting it further out by a certain amount, e.g. 5% [58][7]. Thus, the hypervolume can also be defined as the n-dimensional space dominated by a set of points. Figure 2.2 shows an illustration of the hypervolume in the 3-dimensional space.



| | $x$ | $y$ | $z$ |
|---|---|---|---|
| $a$ | 11 | 4 | 4 |
| $b$ | 9 | 2 | 5 |
| $c$ | 5 | 6 | 7 |
| $d$ | 3 | 3 | 10 |

Figure 2.2: An illustration of the hypervolume of a set of four three-dimensional points [16], in which all objectives are to be maximised. On the left one can see the volumes dominated exclusively by each point, which in this case also form the hypervolume when summed up, while the right table shows the corresponding objective values of each point.

Without any additional measures or calculations, hypervolume encapsulates both convergence and diversity in one value. Only when a front excels in both categories is the hypervolume maximised, which consequently means that the hypervolume is only maximised if and only if the set of points include the points of all non-dominated solutions [33].

The biggest shortcoming of the hypervolume indicator is the high computational complexity of its calculation, which was proven by Bringmann and Friedric to be #P-hard in the number of dimensions, i.e. objectives [17]. However, because of its useful mathematical properties, a lot of research is focused on finding a faster or otherwise more efficient algorithm, which results in a variety of different solutions each with different complexities and varying levels of precision. What follows is a description of three ways of calculating the exact hypervolume of a set of points.

One of the simplest methods for calculating an exact hypervolume is the *Inclusion-Exclusion method* [93], which applies the inclusion-exclusion principle from combinatorial mathematics to the problem. The hypervolume is found by first adding up the n-dimensional volumes dominated by each point, then subtracting volumes encompassed by the intersection of two points, then adding back those dominated by the intersection of three points, and so on. The algorithm continues until all subsets of the set of m points have been considered. Even though the algorithm is simple in its idea, the computational complexity of the Inclusion-Exclusion method is enormous. Since one volume calculation already is at least $\mathcal{O}(n)$ and the volume of every subset of m has to be calculated, the complexity of the whole algorithm is $\mathcal{O}(n2^m)$, i.e. it is exponential to the number of points, making it unusable for most scenarios.

A more efficient way of calculating an exact hypervolume is the *LebMeasure* algorithm [33]. The LebMeasure algorithm works by "lopping off" and adding up exclusively dominated hypercuboids from the total hypervolume one at a time until there is nothing left to remove. First, the hypervolume exclusively dominated by the currently processed point $P$ is considered. If the dominated area is not a hypercuboid, then the largest hypercuboid dominated by $P$ is calculated and "lopped off", while $P$ is replaced with a set of so-called spawn points that dominate the rest of the volume not covered by the aforementioned lopped off cuboid. Otherwise, the dominated volume is simply added to the partial sum and its associated point $P$ will be removed from the calculation. The spawn points are generated by calculating the opposite corner $Q$ of the dominated hypercuboid and replacing one of the objective values of P with those of $Q$, for each objective. For example, consider the points $P = [(6,9,4),(9,7,5),(1,12,3),(4,2,9)]$. The opposite corner of the hypercuboid dominated by the first point $(6,9,4)$ is located at $(4,7,3)$, which means the resulting three spawn point candidates are $S = [(4,9,4),(6,7,4),(6,9,3)]$. From that list of candidates, only those spawn points that dominate an exclusive hypervolume are considered. This procedure continues until there are no more points left to process. Originally, the LebMeasure algorithm was thought to have a polynomial worst case complexity. However, this was later disproved by While [90], in which the lower bound of the complexity was corrected to be $\mathcal{O}(2^{n-1})$, i.e. it is also exponential in the number of objectives. While this makes it unusable for high-dimensional data sets, the LebMeasure algorithm is nevertheless still suitable for cases with a low amount of objectives.

An algorithm that takes a different approach to both presented methods is the Hypervolume by Slicing Objectives (HSO) algorithm developed by While et al [91]. Instead of handling points and volumes in the n-dimensional space, HSO considers one objective at a time. After sorting the set of points in order of the objective being processed, "slices" are created by visiting each point in order. A slice is the hypervolume between the current and the last visited point, whose depth is defined as the distance between those two points. Since each slice is itself an $(n-1)$-dimensional hypervolume, it is calculated recursively and multiplied by the depth of the slice. This process is repeated until all points have been visited. Even though the worst-case complexity of HSO is still exponential in the number of objectives, While et al. found that it outperforms LebMeasure in most cases [91]. Like LebMeasure, HSO should not be used for high-dimensional data though.

Although HSO offers a better performance for most data sets, the ease of implementation of the LebMeasure algorithm and its still comparable performance for the low amount of objectives covered in this work made it a good fit for the calculations in this thesis. If this work was to be extended, e.g. by adding more objectives or by increasing the population size, a better performing solution should be considered.

## 2.2 Evolutionary Multi-objective Optimisation

A commonly utilised technique for multi-objective optimisation problems is the genetic algorithm (GA), which is a subset of the more generic EA. As the name suggests, several of the concepts used in GAs are derived from the Darwinian theory of evolution [21][56]. They operate on a population of organisms, from which an GA selects the best organisms and constructs a new population by applying a set of evolutionary operators, such as mutation or crossover. This makes GAs a great choice for multi-objective optimisation problems, because they generate multiple different solutions per iteration by design [88]. In the following, the basic structure and principles of a GA will be explained. Afterwards, we will examine how the previously introduced concepts used in multi-objective optimisation can be applied to a GA by presenting several different popular MOEAs. Building on that, centralised and decentralised MOEAs and the differences between them will be explored.

### 2.2.1 Genetic Algorithms

The first systems based on the evolutionary theory can be traced back to the 1950s and 1960s, where various computer scientists independently developed such systems [56], from Rechenbergs “evolution strategies” [66] to Fogel et al. [34] developing a technique called “evolutionary programming”. However, a lot of the groundwork for genetic algorithms as they are used today first appeared in 1975 in Holland’s Adaptation in Nature and Artificial Systems [41]. Genetic Algorithms are population-based meta-heuristic optimisation algorithms. They start by randomly initialising a population of organisms $P0$, which is then repeatedly evaluated and modified through a series of genetic operators until a halting criterion is reached. The population of each of these iterations is also called a generation. This process is illustrated in Figure 2.3.

The organisms generated and optimised in a GA are represented by their chromosomes, which the genetic operators are applied to. These chromosomes are each made up of genes, which can be modelled as a list of bits, strings, integers or other data structures, depending on the implementation. Each gene is an instance of a particular allele, e.g. 0 or 1. A chromosome’s composition determines how an organism will behave. For example, one could treat the chromosome as a list of numbers, which are used as indexes for actions in an instruction table. In the following, every step of a GA will be explained in more detail, starting with the initialisation.

#### Initialisation

The initialisation of a population is usually done randomly to ensure a wide spread of solutions and to mitigate premature convergence issues. In some

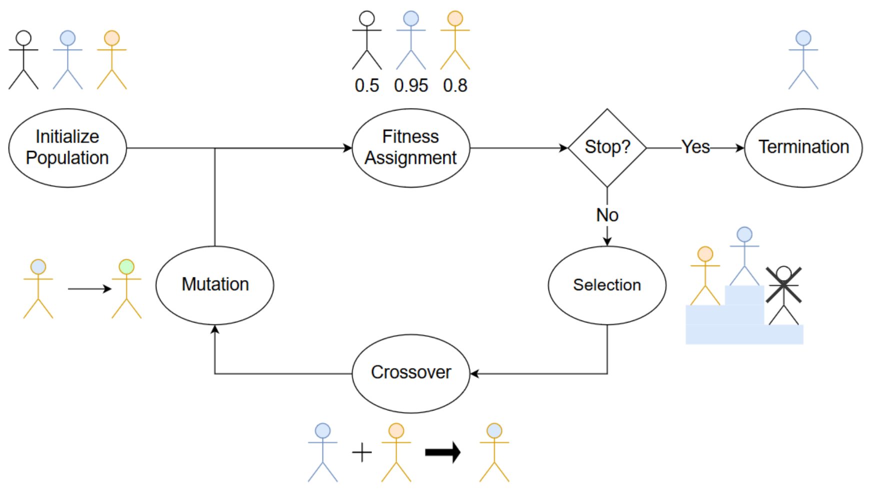


Figure 2.3: The structure of an evolutionary algorithm.

cases, a weight or seed can be applied to steer the initialisation to areas with better chances of finding optimal solutions [56].

### Evaluation

Organisms are then evaluated by a so-called *fitness function*, which measures the quality of a solution and influences whether an organism will survive. Fitness functions are inherently problem-specific. For example, the fitness function for solutions to the Travelling salesman problem (TSP) could be based on the length of that solution's route. The shorter the route of the individual, the better its fitness [49].

### Selection

Following the evaluation, a fitness-based selection algorithm will then decide which organisms will be used as parents for the next generation. Organisms with a higher fitness generally have better chances to be chosen for reproduction than those with a low fitness. Two popular methods of selecting organisms are Tournament and Roulette selection [56]. Tournament selection randomly chooses two or more organisms from the population, where the one with the highest fitness, i.e. the winner of the tournament, will be added to the list of parents. Roulette selection, which is also called fitness proportionate selection, assigns a probability of selection to each organisms based on its fitness relative to the sum of every organisms' fitness.

### Recombination

After selecting the best parents from the current generation, new organisms are created by combining the chromosomes of two or more parents. This recombination of chromosomes is also called "crossover". The simplest form of crossover is the single-point crossover [76], which randomly chooses a point in the parents' chromosomes and combines the part left to the point of one parent with the part right to the point of another parent to form new chromosomes. As an example, consider the two parent chromosomes represented by bit arrays $A = [0, 0, 0, 0, 0, 0]$ and $B = [1, 1, 1, 1, 1, 1]$ with the crossover index $I_c = 3$. The resulting children chromosomes would then be $C = [0, 0, 0, 0, 1, 1]$ and $D = [1, 1, 1, 1, 0, 0]$. These newly created organisms are then added to next generation's population.

### Mutation

Before moving onto the next iteration and thus starting the cycle anew, each chromosome has a chance to trigger a mutation. There are a lot of different types of mutations, the most common being the point mutation, which changes the allele of a single gene, e.g. from 0 to 1. Other types include the insertion mutation, wherein a section of the chromosome is copied from one place to another, or swap mutation, where one gene is swapped with another gene [75].

Mutation helps increasing or at the minimum retaining the diversity of a population by introducing new values and gene combinations. It thus also lowers the chance of a population being stuck in a local minimum forever. Consequently, mutation also makes it easier for the algorithm to find the best possible solutions by allowing it to explore more of the solution space. However, the mutation rate has to be chosen carefully and with respect to the problem at hand. A too high rate prevents the population from converging to an optimal solution, while a too low rate will lead to premature convergence and the population being stuck in a local minimum [56].

### 2.2.2 Multi-objective Selection

In single-objective optimisation problems, like the aforementioned TSP, finding the best fitness value of two or more solutions is usually just a matter of sorting in ascending order and choosing the first or last value, depending on whether the objective should be maximised or minimised. Having said that, in multi-objective optimisation problems the organisms cannot be easily ordered. An organism could be the best in one objective, while having the worst performance in another objective, which is why techniques like partial ordering by Pareto dominance have to be used instead.

#### NSGA-II

One popular algorithm implementing such techniques is the NSGA-II, which was first introduced in Deb et al [26]. It was built on top of the first nondominated sorting genetic algorithm (NSGA) iteration [77], improving on a lot of its predecessor's problems. Whereas the original NSGA has a complexity of $\mathcal{O}(MN^3)$, where $M$ is the number of objectives and $N$ is the number of organisms, this version reduces the complexity to $\mathcal{O}(MN^2)$. Furthermore, the sharing parameter $\sigma_{share}$ was removed in favour of a more dynamic diversity-preserving mechanism. And lastly an elitism operator was added, improving on both the speed and the solution quality of the algorithm [95]. NSGA-II introduces three important operators: Nondominated sorting, crowding distance and the crowded comparison operator $\prec_n$.

Nondominated sorting acts as the algorithm's convergence measure. Similar to other Pareto-dominance based algorithms [35], solutions are assigned a rank $p_{rank}$, which is defined as the number of solutions that dominate $p$. The solutions are then split into fronts by their ranks, creating a rough ordering of the population.

In addition to the rank, each solution is also assigned a crowding distance $p_{distance}$, which acts as a diversity measure. The crowding distance measures how far away neighbouring solutions are from $p$ along each objective, where a higher total distance indicates a higher diversity and thus a better solution. First, the fronts are sorted by each of the organisms' objective values in ascending order. The solution's crowding distance is then determined by summing the distances between the objective values of their left and right neighbours for every objective. The exception are edge cases, i.e. those solutions that don't have a left or right number for any objective, who are instead assigned an infinite crowding distance, since they increase the population's diversity the most. However, if the crowding distance value is needed for further analysis that doesn't handle infinity, the crowding distance of edge cases can also be calculated by doubling the neighbours distance to $p$.

Lastly, both rank and crowding distance are combined to form the crowded comparison operator $\prec_n$, which is defined as

$$\begin{aligned} i \prec_n j \ \text{ if } & i_{rank} < j_{rank} \\ \text{or } & (i_{rank} = j_{rank} \text{ and } i_{distance} > j_{distance}) \end{aligned} \tag{2.5}$$

### Strength Pareto Evolutionary Algorithm (SPEA)

Around the same time, Zitzler and Thiele developed an algorithm based on many of the same principles: the Strength Pareto Evolutionary Algorithm [98]. Instead of storing the whole previous population like the NSGA-II algorithm, SPEA only keeps an archive of the non-dominated solutions that have been found so far. Every generation the non-dominated solutions of the current iteration are moved from the population to the archive. If an older archived solution is dominated by any of the newly added organisms or if it is a duplicate, it will be removed from the archive. This means that the archive always only contains solutions that have been non-dominated across all generations processed so far. Individuals in the archive are then each assigned a fitness value that is equal to the amount of population members dominated by that individual, divided by the population size plus one. On the other hand, the fitness of a population member $p_j$ is calculated by summing the fitness values of every archived solution that dominate $p_j$ plus one. Based on these fitness values, a binary tournament across both archive and population is performed to find the next generation's parents. Since SPEA specifies that these fitness values are to be minimised, archived, i.e. non-dominated, solutions have a better chance at reproduction and survival.

Two years later [96], an improved version (SPEA2) was released by Zitzler et al. To prevent the low selection pressure issues that occurred when the archive only contained one individual, which lead to a drastic performance degradation in the first version, the size of the archive has been changed to a fixed value. If the archive's size falls below that value, dominated individuals from the population are added until it has reached capacity again. Furthermore, the fitness assignment procedure has been adjusted slightly and density information was added to the organisms, which made tournaments between individuals with otherwise identical fitness values less random and helped lessen the problems that arose from configurations with more than two objectives. When compared with other state-of-the-art MOEA algorithms, the SPEA2 showed the best overall performance, together with NSGA-II [96].

#### Multi-objective evolutionary algorithm based on decomposition (MOEA/D)

A method that takes a completely different approach from both presented algorithms is the multi-objective evolutionary algorithm based on decomposition (MOEA/D) developed by Zhang et al [94]. Rather than considering all objectives at once and using measures like Pareto dominance to perform comparisons between solutions like most other MOEAs did at the time, the MOEA/D instead applies the concepts of multi-objective decomposition to an EA, breaking down the problem into a number of scalar optimisation subproblems and optimizing them simultaneously with an EA. Every solution in the population is associated with a specific subproblem through the use of a weight vector. Solutions, whose weight vectors are close to each other, are considered to be part of a neighbourhood. Genetic operators like recombination are only performed inside these neighbourhoods since close weight vectors also imply close optimal solutions to the associated subproblem. When compared with NSGA-II and Multi-objective genetic local search algorithm (MOGLS) [57] on multi-objective knapsack problems, MOEA/D has shown to have a lower computation complexity than both, while producing solutions of similar or higher quality. Furthermore, it was also shown to produce a very uniform distribution of representative Pareto optimal solutions [94].

### 2.2.3 Decentralised Approach

To gain a better understanding of how the integrated information of a swarm optimised by an MOEA evolves over time, two different kinds of GAs will be used and compared throughout the following and the result chapter of this thesis: *Centralised* and *Decentralised* GAs. In centralised GAs, there is one global deciding force that selects the next generation's parents from the whole population pool. On the other hand, decentralised selection occurs on a more local scale, usually by dividing the population into smaller neighbourhoods and selecting the best organisms from each to form the new generation.

**Centralised**, or panmictic [2], selection is typically the default configuration for a genetic algorithm. It always takes the whole population of a generation into account, which means that the chances of a poorly performing organism being considered for reproduction in a fitness-based centralised selection are small, but not non-existent. While this quickly eliminates unsatisfactory solutions, it can also lead to a rapid loss of diversity and, ultimately, to premature convergence. Examples of centralised selection algorithms are those presented in the previous chapter, e.g. NSGA-II or SPEA2.

**Decentralised** GAs, or Spatially structured evolutionary algorithms (SSEAs) [83], offer a way to increase the genetic diversity in the population [27]. Instead of considering the whole population pool during parent selection, the solutions are divided into subpopulations for processing. This allows us to easily divide the work between multiple workstations, processors or threads, increasing the computational performance considerably, while still maintaining the quality of solutions [27].

Since we need a way to define what constitutes a subpopulation, a spatial topology has to be chosen [23]. Depending on the chosen topology, the exchange of solutions between subpopulations can either be performed separately, which is the case with **distributed** EAs, or more organically in the form of neighbourhoods, in which case the EAs are **cellular** [1].

#### Distributed

Distributed GAs are said to be inspired by several different evolutionary theories, including the shifting balance theory in evolutionary biology developed by Sewall Wright [92][46]. They are defined by an insular topology [81][63][20], where each subpopulation represents an island connected via "bridges", which are defined by a migration policy that controls which organisms are allowed to be exchanged between each group. Additionally, this migration policy also controls the islands' topology, the interval in which a migration of organisms

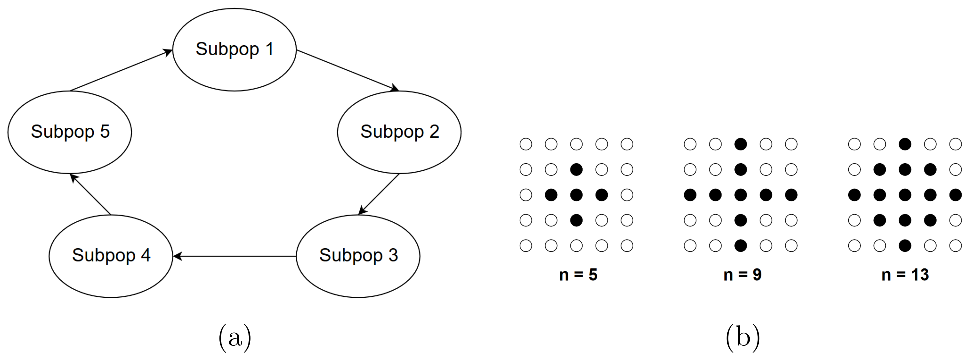


Figure 2.4: Example topologies for distributed (a) and cellular (b) decentralised GAs. Subfigure (a) shows an example of the so-called Ring topology [82], in which sub-populations can only communicate with their directional neighbour. The subfigure on the right side (b) illustrates different possible neighbourhood structures on a two-dimensional grid [23].

occurs and how the migrated individuals are integrated into their new subpopulation [1].

There are a lot of different options when choosing the islands' topology such as hierarchical, hypercube or ring topology [3]. The choice of topology directly influences how much focus will be put on exploitation (convergence) and exploration (diversity) respectively. For example, a fully connected topology would result in good solutions quickly spreading across the entire network, which would lead to a reduction in the diversity of the population and possibly a premature convergence. A topology with few connections also has the advantage of easier parallelisation and scalability [3]. Figure 2.4 (a) shows an example of the ring topology, which limits communication to directionally adjacent subpopulations [82]. It has been shown that a distributed algorithm with such a configuration can achieve better performance in both solution quality and computational speed than a typical sequential GA [42].

One advantage of distributed GAs is the ease of parallelisation, e.g. by simply assigning one workstation or process for each island, while a separate workstation controls the migration policy. Since subpopulations only seldom communicate with each other, individuals in these islands are less restricted in the direction they want to evolve in. Due to this limited communication, good solutions spread slower, which means that the diversity of a distributed GA can be maintained more easily than with a classical GA [64][51]. Apolloni et al. [6] found that their custom island-based distributed GA achieved better performance than eight sequential EAs when applied to the CEC 2005 test suite [80].

### Cellular

The structure of cellular GAs is, contrasted with island-model GAs, more fine-grained [38]. Instead of dividing the population into smaller subpopulations, cellular GAs instead position organisms onto a grid, where individuals can only interact and compete with each other if they are part of the same neighbourhood [62], i.e. if the amount of "steps" between their positions on the grid is smaller than a given neighbourhood size. Figure 2.4 (b) shows an example of how different neighbourhood sizes might be realised for a 2D grid. Neighbourhoods can overlap each other, which means that an organism can be part of more than one neighbourhood. Compared to GAs with unrestricted interactions between organisms, which are also called panmictic GAs [2], cellular GAs have a much lower selection pressure, resulting in a slow diffusion of solutions across the grid and thus a more stable population diversity [62]

One popular multi-objective cellular GA is the Multi-objective cellular genetic algorithm (MOCell) algorithm developed by Nebro et al [60]. Similar to other previously mentioned algorithms, MOCell keeps an external population containing the best solutions (according to the crowded comparison operator $\prec_n$) encountered so far, from which a few randomly selected solutions are moved into the current population in each iteration. Genetic operators like recombination and mutation are applied on a per-neighbourhood basis. If an offspring performs better than their parent, the former will replace the latter in the next generation. This next generation is saved in a separate auxiliary population during this process, which means that the original population remains unchanged from the genetic operators. After all neighbourhoods have been considered and the archive has been updated with the newly created offspring, the changes in the auxiliary population are applied to the real population simultaneously, i.e. the original MOCell implementation is a synchronous cellular GA [59].

When compared to NSGA-II and SPEA2, the solutions produced by the original algorithm achieve a similar or superior performance in most of the problems of the WFG catalogue [60]. The authors further improved the performance of the algorithm in [59], in which Nebro et al. compared six different MOCell variations with each other, by slightly changing the archive solution merging process and making the population update process asynchronously, i.e. updating the solutions sequentially. Another algorithm based on MOCell is CellDE [28], which replaces the original genetic operators with the reproductive operators of differential evolution [78].

To sum up, it can be said that SSEAs offer various advantages over sequential GAs. On the one hand they provide a more robust way of finding a diverse set of Pareto-optimal solutions, and on the other hand they also often offer a better performance in both computation time and solution quality [3] than conventional GAs. However, there is also higher effort, e.g. building the communication layers or finding the right cutoff point for distributions, associated with using an SSEA, which means that they are not necessarily the best option for every optimisation problem.

The synchronous cellular genetic algorithm used in this thesis will be presented in the third chapter.

## 2.3 Integrated Information Theory 3.0

One of the most prominent leading theories in the field of machine consciousness is the Integrated Information Theory 3.0 developed by Tononi et al [84][85][8][61]. It provides a mathematical framework, which allows one to evaluate the consciousness of a system in both quantity and quality, effectively making consciousness testable. Contrary to a lot of other theories in the field that take the physical properties of a system and attempt to arrive at consciousness, IIT 3.0 identifies the fundamental properties of consciousness, from which the authors derive the essential traits of a conscious system. The five basic properties of consciousness defined by the authors, which are also called axioms, are: Existence, Composition, Information, Integration and Exclusion. For each axiom a corresponding postulate is inferred, which collectively describe how a physical mechanism, e.g. a neuron or a logic gate, must be configured to allow them to give rise to consciousness. By applying these axioms and postulates on the level of both mechanisms and systems, the theory finally defines an experience as a MICS [61][9]. This structure essentially describes the quality of an experience, while its integrated information, which is measured in $\Phi^{Max}$, describes its quantity.

In the following section, the groundwork for the machine consciousness related part of this thesis will be laid. First, the current iteration of the integrated information theory (IIT 3.0) will be explained in more detail. This is followed by a discussion about IIT's limitations and criticisms. Finally, the underlying brain structure of the organisms analysed in this thesis, the Markov Brain, will be presented.

### 2.3.1 Axioms

Tononi et al. define axioms in the context of IIT as self-evident truths about consciousness, i.e. undeniable statements that, "with Descartes, do not need proof" [61].

The first axiom, the **Existence** axiom, simply states that consciousness exists. Each experience exists intrinsically, independent from outside observers.

The **Composition** axiom implies that every experience consists of several different elementary components in various combinations. For instance, the experience of a blue car driving by consists of the combination of seeing the colour blue, the car, the car moving from left to right and possibly feeling the wind produced by said movement.

The **Information** axiom states that every experience is informative, which means that it is distinct from all other possible experiences. Even an experience consisting of pure darkness contains information, since it differs from other scenarios like the car driving by.

The **Integration** axiom implies that experiences are also integrated, i.e. they are strongly irreducible to non-interdependent components. One cannot take away a part or component of the experience without changing its information.

Finally, the **Exclusion** axiom states that every experience excludes all other experiences. It is not possible for experiences to overlap each other partially, for every possible time and place only one experience can have its full content. In other words, experiences have a specific spatio-temporal grain [9].

## 2.3.2 Postulates

In the following section, the postulates of IIT will be applied on individual mechanisms. With the resulting findings we will then apply the postulates at the level of systems of mechanisms. The concepts shown in both sections will then allow us to quantify and analyse a system's ability to give rise to consciousness.

### At the mechanism level

The **Existence** postulate acts as a simple starting point in finding a conscious entity. It simply states that mechanisms in a state exists and that a set of mechanisms form a *system*. "Mechanism" in this case meaning anything having a causal role within a system, e.g. a neuron in a brain or a logic gate in a computer.

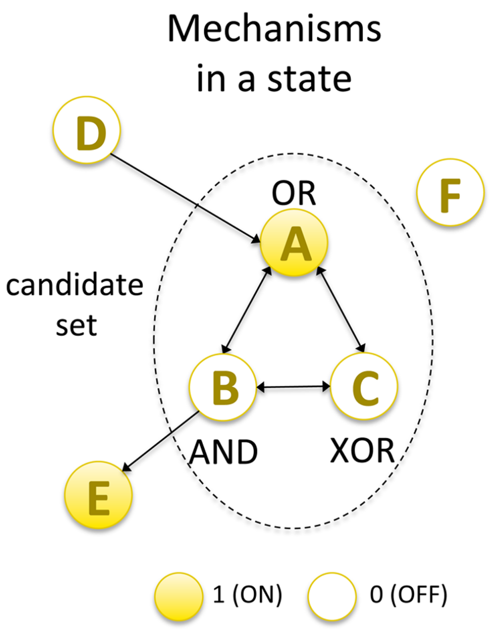


Figure 2.5: An example mechanism in the state $ABCDEF = 100010$ [61].

In this work, systems are built from combinations of discrete logic gates. Figure 2.5 shows an example of such a system [61]. The nodes $A$, $B$ and $C$ each specify their own logic gate mechanisms, $OR$, $AND$ and $XOR$ respectively, which are connected through inputs and outputs. These three form a candidate set for integrated information analysis, which is indicated by the dotted line, while $D$, $E$ and $F$ can be thought of as external background conditions. In this case $D$ is the only external node that has an impact on the candidate set, since its output is directly connected to $A$, whereas the other two nodes are only connected through their inputs or not at all. The node colouring indicates the state the nodes are in, where yellow means 1 and white means 0, which means the state depicted in Figure 2.5 is $ABC = 100$ with background conditions $DEF = 010$.

Building on the existence postulate, the **Composition** [61] postulate states that these mechanisms can be combined into high-order mechanisms. In the aforementioned figure, the candidate set's elementary, or first-order, mechanisms are the nodes $A$, $B$ and $C$. Combining two of these yields a corresponding second-order mechanism ($AB$, $AC$ or $BC$), while integrating all elementary mechanisms creates the third-order mechanism $ABC$. The set of all possible combinations of such a candidate set is called a *power set.* When analysing a specific mechanism that doesn't include all nodes of a mechanism, the left-out nodes are treated as independent noise.

The **Information** [61] postulate requires that a mechanism's state $S$ has to influence the state of a system from its own intrinsic perspective for it to generate information and thus be able to give rise to consciousness. The probability distribution of all possible past states given the current state $S$ is called the *cause repertoire.* This distribution is compared to the unconstrained distribution of the system's past states, where the distance between both indicates the amount of *cause information (ci)* the mechanism generates. Similarly, the constrained distribution of future states, the *effect repertoire*, is compared with the unconstrained effect repertoire to calculate the mechanism's *effect information (ei).* Typically, the distance measure used is the Earth Mover's Distance (EMD) [69]. A mechanism's combined generated information, or *cei*, is then defined as:

$$cei = min(ci, ei) \tag{2.6}$$

To illustrate why the minimum of both measures was chosen, consider a mechanism $X$ that constrains the system's past, but not the future states. Since the result of $X$ doesn't have an influence on the system, it doesn't make a difference from the intrinsic perspective of the system itself, even though it conveys information about the system's past states for an external observer. The other way around, consider a mechanism $Y$ that constrains the future, but not the past states. In that case, what the system does doesn't have an effect on the mechanism, which means it cannot at all control the state of $Y$, thus decreasing the overall information to zero.

The **Integration** [61] postulate states that only mechanisms that generate integrated information can give rise to consciousness. Removing any part of such an integrated mechanism results in a change or loss of information. Subsequently, mechanisms that contain redundant or unused parts cannot specify integrated information. Put differently, they are irreducible with respect to their information.

To calculate the integrated information $\phi$, the *cause-effect repertoire*, i.e. the probability distributions of past and future states, of every possible partition of the mechanism is determined. The partitions whose cause or effect repertoire are closest to the one of the whole mechanism are called *Minimum Information Partition (MIP).* The distance $D$ between the MIP's distribution and the distribution of the whole system is then calculated separately for the cause and effect repertoires. Similar to the way a mechanism's information is defined, the integrated information is defined as the minimum of both distances. Mathematically, $\phi$ can be defined as:

$$\begin{aligned}
\phi_{cause}^{MIP}(M^p|M^c = S) &= D(p(M^p|M^c = S)||p(\frac{M^p|M^c = S}{MIP})) \\
\phi_{effect}^{MIP}(M^f|M^c = S) &= D(p(M^f|M^c = S)||p(\frac{M^f|M^c = S}{MIP})) \\
\phi^{MIP}(M^{p,f}|M^c = S) &= min(\phi_{cause}^{MIP}(M^p|M^c = S), \phi_{effect}^{MIP}(M^f|M^c = S))
\end{aligned} \tag{2.7}$$

where $M$ is the mechanism we want to analyse and $S$ is the current state of $M$ [61].

The **Exclusion** [61] [9] postulate at the level of individual mechanisms states that only a mechanism's maximally irreducible cause and effect are considered, whereas other causes and effects are excluded from analysis. To find the so-called *core cause* and *core effect* of a mechanism, the integrated information of all possible partitions, i.e. its power set, is calculated. The partition with the highest $\phi$ value is considered maximally irreducible. As before, the core cause and core effect are determined separately. Mechanisms that specify both a core cause and a core effect, also called the *Maximally Irreducible Cause and Effect (MICE)*, form a (core) concept. The integrated information of those core cause and effects is called $\phi^{Max}$. While the MICE describes the quality of an experience or concept, $\phi^{Max}$ specifies its quantity, i.e. how strongly it is integrated.

## At the system level

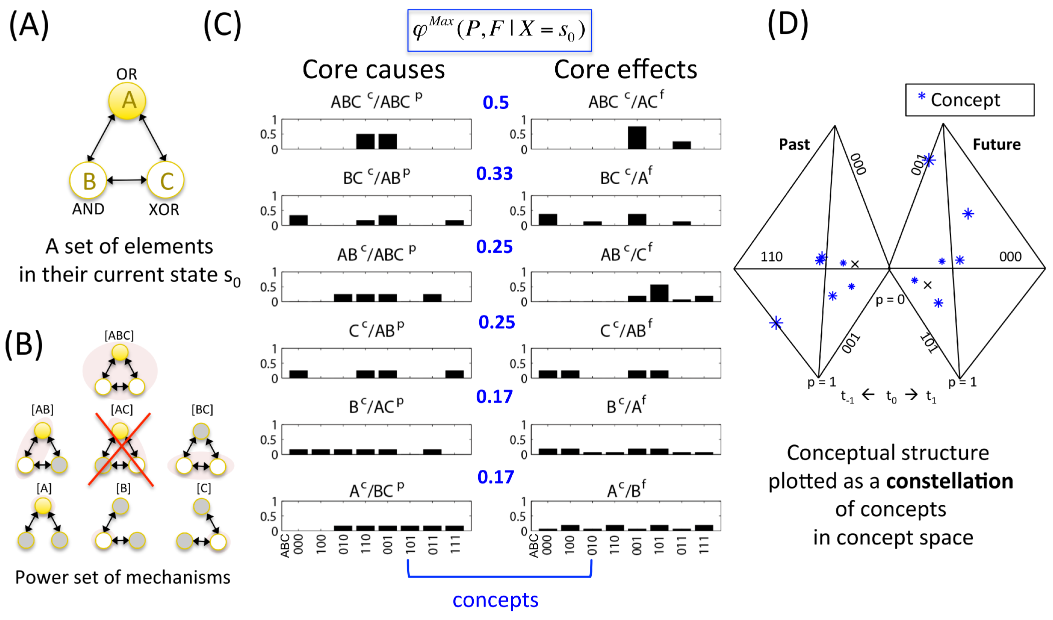


Figure 2.6: A conceptual analysis of a set of elements *ABC* [61]. (A) depicts the structure of the brain. Every permutation of *ABC* (shown in (B)) that produces cei greater than 0 specifies a concept. The cause-effect repertoires of said concepts are plotted in (C). (D) shows these same concepts mapped onto the so-called *concept space*, which is also called a *conceptual constellation*.

Similar to its description at the level of a single mechanism, the **Information** [61] postulate at the system level states that information is only generated by sets of “differences that make a difference” [61], which in this case refers to a constellation of concepts. This constellation is constructed by iterating over all possible mechanisms of the power set of the candidate set and computing the integrated information $\phi^{Max}$ for each of them. Those that generate non-zero integrated information each specify a concept. Together, the concepts form a conceptual structure.

Figure 2.6 (D) shows such a structure represented in the so-called concept space. The axes of the concept space each map to a possible past or future state of the mechanism. For instance, three nodes would allow for eight different states and thus sixteen axes (eight for each past and future). The position of a concept on those axes is determined by the corresponding probability specified by the concept's cause-effect repertoire, while the size of the dot is determined by its integrated information. Visualisations of the concept space (see Figure 2.6 (D)) typically separate past and future states from each other and restrict them to three dimensions or axes each to make it easier to read.

To quantify the amount of information generated by a system, the Conceptual Information (CI) is calculated. Similar to the cause-effect information at the mechanism level, CI is defined as the minimum distance from the constrained to the unconstrained repertoire $p^{uc}$, corresponding to the null concept, i.e. a concept that specifies nothing.

The **Integration** [61] postulate at the system level states that only systems whose mechanisms form an irreducible conceptual structure are able to give rise to consciousness. Analogous to the postulate at the mechanism level, excluding any mechanism from an integrated system would result in a change of conceptual information. Accordingly, if a system can be partitioned in such a way that the information is unchanged, it cannot be integrated.

To find a system's integrated information $\Phi$, the system is partitioned unidirectionally until the partition with the least amount of impact on the system's conceptual information, i.e. the MIP, has been found. $\Phi$ is then defined as the EMD between the conceptual constellation of the whole system and the MIP.

Lastly, the **Exclusion** [61] [9] postulate at the system level states that only MICSs [9] can lead to consciousness. Furthermore, mechanisms can only be part of one such structure, i.e. elements cannot overlap each other. The set of elements in a system that generate a local maximum of integrated information $\Phi^{Max}$ constitute a *complex.* To find such a *complex*, the integrated information $\Phi$ of each member of the power set of the system, excluding the single-order elements since they cannot be partitioned any further, is determined. The candidate set with the highest $\Phi$ value is called the *major complex*. Any additional complexes that the system consists of are called *minor complexes*. However, as mentioned previously, elements inside one complex cannot be part of another one. For instance, a system of five nodes $ABCDE$ with a major complex $CDE$ cannot form a second complex that contains either $C$, $D$ or $E$. A minor complex $AB$ would still be possible though.

### 2.3.3 Discussion

The Integrated Information Theory 3.0 provides a strong mathematical framework for the exploration and evaluation of consciousness. However, there are also some limitations and criticisms other researchers in the field of consciousness have brought up. For instance, Manzotti [52] argues that the phi value might only measure the information processing properties of a system rather than a subjective experience or consciousness [67]. Furthermore, Seth et al. [74] showed that it is possible to nearly arbitrarily inflate the phi value of a fully-connected neural network with a suitable set of synaptic weights. This doesn't necessarily discredit the theory in itself, but it would mean that such a network would at some point exhibit a substantial amount of consciousness, which leads some to doubt whether the theory can actually share any light on the nature of consciousness at all [73]. It has also been argued that IIT lacks some basic properties, like structural coherence and an organisational invariance, that are required for a theory about consciousness to be considered well-formed [19]. Another very recent paper by Bayne [10] is sceptical about the axiomatic nature of IIT. In it he claims that the theory's axioms don't necessarily correspond to essential properties of consciousness and even argues that we as humans are ill-equipped to identify these properties due to our limited access to different forms of consciousness.

Aside from these criticisms, there are also some practical problems that limit the use of IIT. Determining the integrated information of a larger network, such as the human brain, is simply not feasible since the computational complexity is too high. However, there already exists some preliminary research focused on finding a computationally efficient way to approximate the integrated information of such a network [36] [5], which could help alleviate these problems. At the moment of writing this thesis there also doesn't yet exist a gold standard, such as a human brain's integrated information, to compare calculated phi values to, which makes the interpretation of results hard [36]. In the future, such a gold standard could be established by applying the aforementioned approximation algorithms on large-scale brain models such as the ones reviewed in [18] and [22].

## 2.3.4 Markov Brains

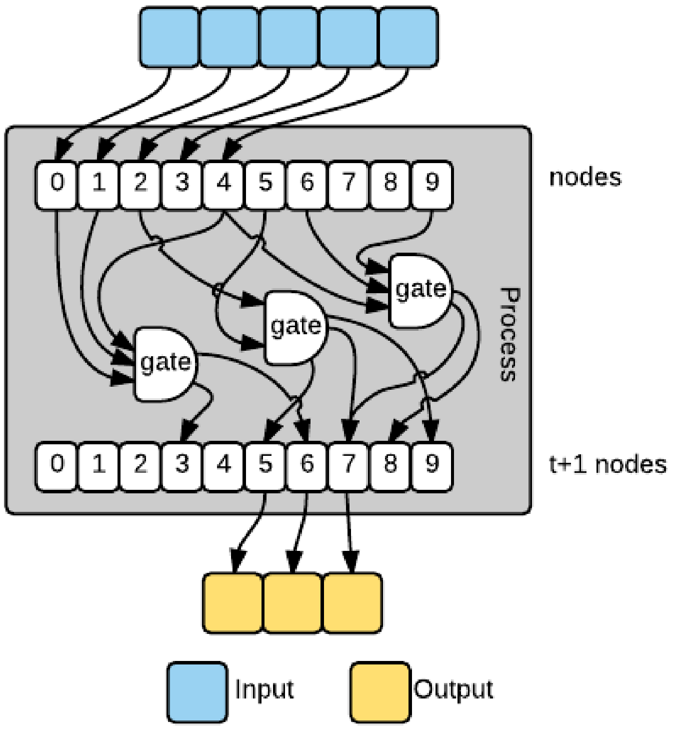


Figure 2.7: An example Markov Brain with 5 input, 3 output and 2 hidden nodes [13]. The first row of nodes inside the brain correspond to the time step $t$, while the bottom row represent the nodes at state $t+1$. The three gates between both time steps each implement a specific function and are essentially responsible for the functionality of the whole brain.

One of the possibilities of modelling an organism in an IIT-compatible way is the Markov Brain (MB). MBs are a subcategory of evolvable Artificial Neural Networks (ANNs) [70], which consists of different components, or gates, linked together in various ways [39]. Rather than layering the components and thereby restricting the possible connections between nodes, any component can be connected to any other component. The components are divided into input nodes, output nodes and hidden nodes. Input nodes receive their values from the environment a Markov Brain agent is living in, i.e. updates caused by connections to input nodes will be overwritten in the next time step. If needed, one could read them before the environment writes into them though. While hidden nodes are used to perform additional auxiliary tasks, e.g. storing the previous update's results, the outputs of a brain, i.e. the values of its output nodes, are used to control the agent's actions. A visualisation of a MB can be seen in figure 2.7.

A MB can be encoded by a genome that controls the topology of the network and the function of its components. Evolving such a genetically encoded MB can then be achieved by performing genetic operators on its genome, which also changes the network. Alternatively, one could also perform these operators directly on the network, thereby directly changing the wiring of connections between components and their behaviour. These kinds of MB are called directly encoded Markov Brains. However, most implementations use the genetically encoded variant, since directly encoded MBs are considered to be too fragile [39].

The genome of a Markov Brain is implemented as a vector of up to 20000 randomly initialised elements, or *sites*, that are each represented by a number. This genome is then interpreted from left to right. First, the interpreter searches for a specific sequence of numbers, or *start codon*, that indicates the start of the *gene*. The numbers used as a start codon depend on the gate type of the network. For instance, deterministic gates use the codon $(42, 212)$ to mark the start of the gene, while probabilistic gates use $(42, 213)$. Once the starting codon has been found, the numbers following that marker will then be interpreted according to a specific scheme to form a gate. The first two sites respectively specify the amount of inputs and outputs of the gate. The next chunk determines how the gate is connected to the nodes of the brain, followed by additional values that define the gate's functionality.

Components used in a Markov Brain can be of one of many different types of gates [39]. Hintze et al. provide a set of already implemented gate types in their paper and the accompanying framework *MABE*[15]. However, there is technically no limitation to the functionality of a gate. The first developed gate types were the deterministic and the probabilistic gates. Deterministic gates work similar to logic tables in that they operate on binary values and map an input to a specific output. Such a gate can have an arbitrary amount of input and output values. However, they are usually limited to at least one and less than five values each. An example of a deterministic gate with one input and output each would be a gate that takes its input value (0 or 1) and simply negates it, emulating the behaviour of a *NOT*-gate. Probabilistic gates on the other hand map their input values to stochastic values, i.e. values between 0 and 1, that are interpreted as probabilities of the output states. It is apparent that deterministic gates are a special case of probabilistic gates where all values are either 0 or 1. Other gate types presented by Hintze et al. include the ANN gate that acts as a single-layer artificial neural network, the Threshold gate, which accumulates its inputs and only outputs a one once the accumulation exceeds a certain threshold, and the Feedback gate, which incorporates external positive and negative feedback into the probabilities of each input-to-output mapping.

Since their introduction in [39], Markov Brains have been used in various publications, including some that furthered the research on the Integrated Information Theory. Since Markov Brains allow connections between any component, meaning one can easily create an integrated system inside a MB, and feature state-based behaviour, they are a suitable candidate for IIT analysis. For instance, in [44] the authors analysed how the amount of integrated information of an agent equipped with a MB correlates to their fitness, which was determined by measuring how well that agent could navigate out of a maze-like room. Joshi et al.'s results indicated that the minimal complexity of an agent's brain is proportional to its fitness, noting that the integration levels above that minimum may deviate substantially from said proportionality. Edlund et al. [29] investigated whether the IIT can accurately provide a measure of complexity of an agent's brain for navigation scenarios. While initially similar, the authors differ from the previously mentioned paper by focusing on agents that consist of Markov Brains with an internal memory. [4] explored a scenario, in which agents with a Markov Brain have to solve a Tetris-like game, i.e. falling blocks of different sizes and shapes have to be arranged in a certain way. Albantakis et al. found that the agents whose tasks required more of the brain's internal memory displayed a higher amount of integrated information over the course of an evolutionary algorithm.

# 3 Experiment

In the following chapter we will describe the experiment and how it is structured in more detail. First, an overview of the general setup of the experiment, which includes the level layout, the animat design and details about the multi-objective selection process, will be given. Afterwards, the different experimental configurations and specifics about other parameters will be presented. Lastly, we will present implementation details of the experiment.

## 3.1 Setup

This section will provide an overview over the multi-objective simulation of this thesis. First, the spatial layout, a two-dimensional grid, will be described. Then, the design and functionality of the organisms will be explained. Lastly, the last two subsection focus on details of the multi-objective optimisation, namely the identified objectives and the neighbourhood selection algorithm developed for this thesis.

### 3.1.1 Layout

The foundation for the generation of the organisms we want to analyse is an evolutionary simulation, whose layout was inspired by the work by Koenig et al[47] and Fischer [32]. Figure 3.1 shows the layout used in this thesis. It consists of a 2D world on an equilateral Cartesian grid of size 32, where two "rooms" of the same size are connected via a 7 block wide gateway. The rooms are surrounded by 1-block thick walls with the wall between them being 3 blocks thick, i.e. the left rooms' dimensions are 14x30, while the right room is one block thinner.

These rooms each contain 36 starting points for organisms, which each have at least two "normal" grid block between them and the next starting slot.

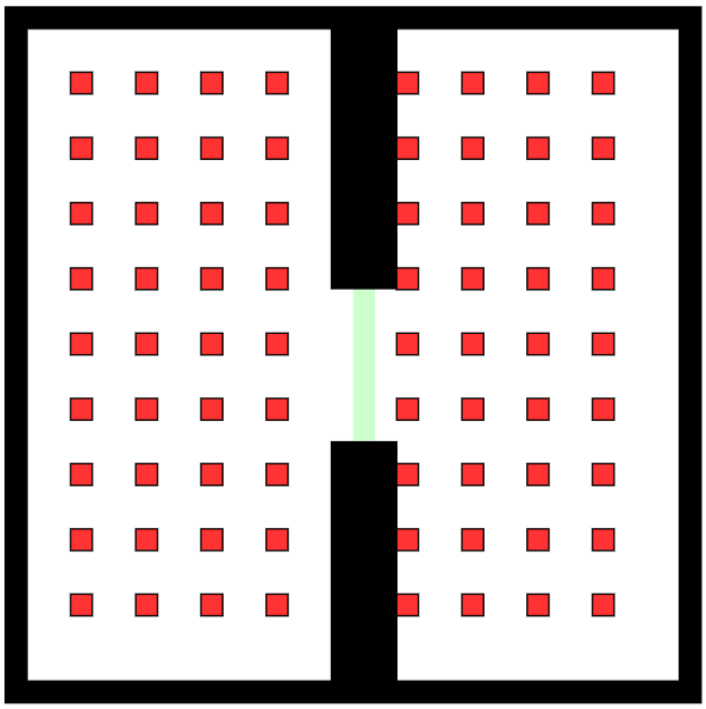


Figure 3.1: The grid the organisms are placed on [32]. Black cells mark the non-passable walls, red cells are starting positions, green cells make up the goal and white cells are regular passable cells.

For each generation, up to 72 organisms are being distributed across these two rooms. To avoid initial positioning playing too large of a role for an organism's fitness and thus survival, the order in which they are assigned is calculated in a non-linear fashion and each generation is repeated multiple times.

The pseudo code of the organism placement algorithm is shown in 1. Each unused position $P$ is assigned a weight corresponding to the amount of organisms placed on the same row or column as $P$. To increase the spread of organisms across the grid, the position with the lowest weight is chosen. If there are multiple grid points with the same weight, the algorithm will fall back to simply randomly choosing from that subset of points. Compared to placing the organisms linearly, this positioning strategy can significantly reduce the amount of collisions, since it allows the organisms to move more freely, especially for simulation runs where $N$ is smaller than 72, i.e. when not every slot is filled.

**Algorithm 1** Organism Placement

```
1: function GETNEXTPOSITION($usedPositions, availableSlots$)
2:     $u \leftarrow unusedPositions(usedPositions, availableSlots)$
3:     $weightMap \leftarrow \{\}$    ▷ the weight map indicates how many agents are placed on the same row or column as the unused position. A lower value is more desirable since it leads to an increased spread of the population.
4:
5:     for $unused$ in $u$ do
6:         $posOnSameRowOrCol \leftarrow 0$
7:         for $used$ in $usedPositions$ do
8:             if $unused.x == used.x$ or $unused.y == used.y$ then
9:                 $posOnSameRowOrCol \leftarrow posOnSameRowOrCol + 1$
10:            end if
11:        end for
12:
13:        $weightMap[unused] \leftarrow posOnSameRowOrCol$
14:    end for
15:
16:    return $getPositionWithLowestWeight(weightMap)$
17: end function
```

### 3.1.2 Animat Design

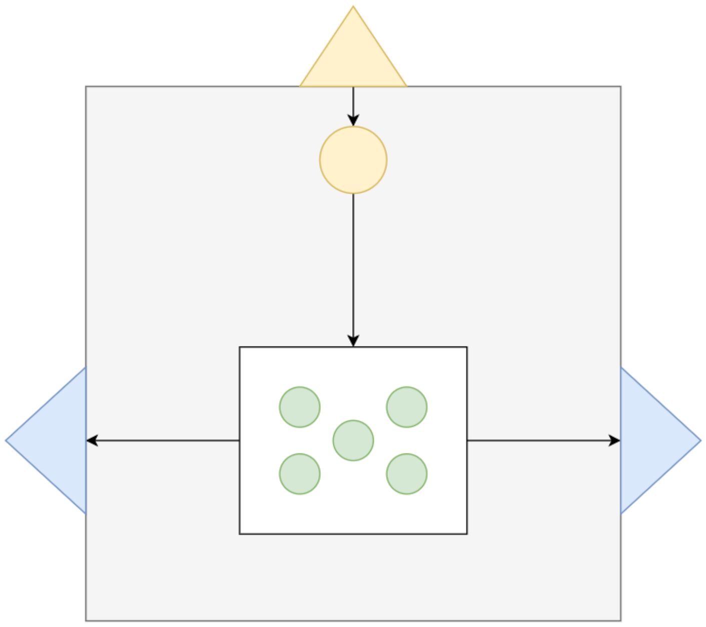


Figure 3.2: The design of the agents used in the simulation. The yellow symbols are the two inputs, which are generated by the front sensor of the organism. The green circles represent the five hidden nodes and the blue triangles symbolise the outputs. Recreated from [32].

The organisms, or animats, are each controlled by a Markov brain. The brain structure chosen for this thesis consists of two inputs, five hidden nodes and two output nodes to maximise the probability of an integrated mechanism in the system [32] while limiting the amount of possible states. This design, which is also shown in figure 3.2, corresponds to the $\gamma$ design in [32].

Located at the front of the organism, the input sensor measures two distinct values, resulting in two sensor values each corresponding to one of the input nodes. While the first value measures whether the given grid point is valid, which in this case means that the point is not out of bounds and also not a wall, the second sensor value indicates whether that field already hosts an agent.

The output nodes form a binary value which is mapped onto a set of four possible actions. These actions are all applied relative to the current position of the animat and the direction it is facing. If both output nodes are off, i.e.

0, the animat will do nothing in this time step. Both nodes being set to 1 results in the animat moving forward one step, provided it is possible to do so. However, if only one of the nodes is set to 1, the organism will turn either left, when the first node is 1, or right, when the second node is 1.

There is no direct active communication between animats outside of sensing a neighbouring organism through one of the inputs.

### 3.1.3 Objectives

Three objectives have been chosen for the simulation's multi-objective optimisation: Maximising *Gate Passages* and minimising *Collisions* and *Movement Penalties*.

**Gate Passages** measure how often an organism has passed through the gate connecting the two rooms. Since that measure alone caused a few behavioural problems, some additional preconditions were introduced. The first condition is that the organism has to walk in a straight line through the gate to score, which acts as a counter measure against standing still on the gate line and scoring points by not moving at all. To fulfil this condition, the organism has to touch one non-goal field directly before and after being on a goal field. Additionally, a timer was added, which is set to a fixed value after successfully performing a gate passage and counts down by one each time step. While that timer is greater than zero, all gate passages performed by an organism are ignored. This makes strategies like only moving the bare minimum of a straight line to satisfy the first condition less viable and incites the organisms to make more use of the two rooms.

**Collisions** happen when an organism tries to move forward by one step, but is blocked by another organism already standing on that field. Only the organism who initiated the move, and is thus responsible for the collision, is penalised, the other animat is spared.

Using only these two measures resulted in a lot of organisms standing still in a corner to try and minimise the collisions by minimising how much the animat moves. Since this effectively meant that one of two objectives was perfectly optimised, the selection algorithm considered their fitness values superior to most other animats, which lead to more and more organisms standing still the longer the simulation was performed. To counteract this trend, the movement penalties were introduced.

**Movement Penalties** quantify how often an organism performed a non-desirable move. These consist of standing still and rotating on the spot for an extended period of time, which in this case means three time steps. With this additional objective, the organisms are incentivised to move more freely, resulting in less animats standing still during the simulation.

### 3.1.4 Selection Algorithm

To gain a better understanding of how the integrated information of the population evolves over the course of a multi-objective evolutionary algorithm, multiple different experiment settings were used (see 3.2). One of these parameters is the *selectionAlgorithm* parameter, which controls the algorithm used in the selection part of the evolutionary process. There are two different options: a centralised algorithm based on the previously established NSGA-II algorithm by Deb et al. [26] and a decentralised synchronous cellular neighbourhood-based selection algorithm, which was developed as part of this work. In the following, the neighbourhood solution will be explained in more detail.

As opposed to the centralised selection process of the NSGA-II algorithm, the neighbourhood solution operates in a decentralised way. Instead of selecting possible parents from a single, global pool of organisms, competition occurs only between those individuals that are near each other. A neighbourhood of organism $o$ of a population $P$ is defined as

$$N_o = \{x | x \in P \wedge d(o,x) \leq d_{max}\} \tag{3.1}$$

where $d(o,x)$ is the Manhattan Distance [48] between the last position of organisms $x$ and $o$ on the grid and $d_{max}$ has been set to 2, i.e. it is a *von Neumann neighbourhood* with r = 2.

The neighbourhood selection algorithm can be divided into two alternating stages, the *copy stage* and the *mutation stage.* In the copy stage, the neighbourhood of every solution $o$ is scanned for solutions dominating it. If it is not dominated by any other solution, $o$ will not be changed and will stay in the population. Otherwise, the best dominant solution, which is determined by their non-dominated rank and crowding distance, is copied onto $o$'s place and replaces it in the next generation. The pseudocode for the copy assignment is also shown in 2.

**Algorithm 2** Copy Assignment

```
1: function ASSIGNCOPY($p, n$) ▷ where p is the population and n is a list of neighbourhoods
2:     $assignments \leftarrow \{\}$
3:     for $i = 0$ to $|p|$ do
4:         $dom \leftarrow \{x | x \in n \land x \prec_n p_i\}$
5:         if $|dom| == 0$ then
6:             $assignments.push(p_i)$                          ▷ no copy
7:         else if $|dom| == 1$ then
8:             $assignments.push(dom.first)$    ▷ dominating org. replaces $p_i$
9:         else
10:            $dom \leftarrow dom.fastNonDominatedSort()$         ▷ sort by $\prec_n$
11:            $assignments.push(dom.first)$          ▷ best org. replaces $p_i$
12:        end if
13:    end for
14:    return $assignments$
15: end function
```

Rather than directly producing a mutated offspring, mutation is delayed until the mutation stage. Every other generation, all organisms have a chance to be replaced with mutated versions of themselves, regardless of their neighbourhood status.

When using the neighbourhood-based approach, organisms that are more active have a better chance at spreading their genetic information, because they get in more contact with more organisms. Due to the slow diffusion of solutions we hope to achieve a higher diversity of solutions than the NSGA-II approach.

## 3.2 Data Analysis

For our data analysis we considered 24 different parameter combinations. To avoid randomness playing too much of a role, which could possibly distort the evaluation results, every configuration is repeated 31 times. The simulation is set to run for 5000 generations, with a snapshot of the population being taken every 100th generation, which results in around 1.7 million database rows. Additionally, the objective values and identifiers of the fronts of every 100th selection process and corresponding supplementary front meta-data like the hypervolume are stored as well, resulting in another 38000 rows. Taking a snapshot more often, possibly even at every generation, in hopes of a higher accuracy was at first considered, but quickly discarded. Firstly, it would have had a large impact on the performance of the simulation, since the calculation of the hypervolume takes up a lot of time, and secondly it would have resulted in way too much data.

Every generation consists of 2000 time steps. In each time step, every organism is updated once, allowing for one action per time step. This update is computed sequentially, which means that if two organisms would go for the same grid field, the first one to be updated would take precedence over the other one. In fact, the second organism would not even know that the field was empty, since their input sensor would not report the state of the field until it is that organism's turn to be updated.

Due to the large amounts of organisms and the high computational cost associated with each IIT calculation, we only consider the first two fronts of each snapshot for IIT-related analysis. Similar to [4] and [32], we apply five different measures. $\Phi^{Max}$ specifies the amount of integrated information generated by the Main Complex (MC) of the brain. $\Phi^{Max}_{concepts}$ and $\Phi^{Max}_{elements}$ respectively measure the amount of concepts and mechanisms contained in the MC. Lastly, $\sum \phi^{Max}$, i.e. the sum of the highest $\phi$ values of each mechanism, and $\sum \phi^{Max}_{concepts}$, the number of concepts specified by all mechanisms in the brain, are stored as well. In contrast to the first three, these last two measures are not limited to just the MC, but provide information about the whole brain of the animat.

The different parameter values that make up the 24 experiment settings are shown in Table 3.1. The first parameter, *selectionAlgorithm*, controls which evolutionary optimisation is performed during the simulation and can be set to either $NSGA2$ or $Neighbourhood$. The second parameter *nAgents* determines the amount of organisms in the simulation and thus the population size. For our experiment we chose to use population sizes of 72, 36, 18 and 1 or 100%, 50%, 25% and 1%, respectively. Lastly, the *mutationRate* parameter controls the probability of a point mutation occurring during recombination, which are set to 0.001 (low), 0.005 (medium) and 0.01 (high) in this thesis. Other mutation rates offered by MABE are fixed for every experiment.

The NSGA-II algorithm used in this thesis uses a tournament size of five for every experiment setting. For recombination, two parents are used to create a single offspring for the next generation, except for experiments with a population size of 1, where only parent is used.

| selectionAlgorithm | popSize | mutationRate | Identifier |
|---|---|---|---|
| NSGA-II | 1 | low | $\alpha_1^{low}$ |
| NSGA-II | 1 | medium | $\alpha_1^{medium}$ |
| NSGA-II | 1 | high | $\alpha_1^{high}$ |
| NSGA-II | 25% | low | $\alpha_{25}^{low}$ |
| NSGA-II | 25% | medium | $\alpha_{25}^{medium}$ |
| NSGA-II | 25% | high | $\alpha_{25}^{high}$ |
| NSGA-II | 50% | low | $\alpha_{50}^{low}$ |
| NSGA-II | 50% | medium | $\alpha_{50}^{medium}$ |
| NSGA-II | 50% | high | $\alpha_{50}^{high}$ |
| NSGA-II | 100% | low | $\alpha_{100}^{low}$ |
| NSGA-II | 100% | medium | $\alpha_{100}^{medium}$ |
| NSGA-II | 100% | high | $\alpha_{100}^{high}$ |
| Neighbourhood | 1 | low | $\beta_1^{low}$ |
| Neighbourhood | 1 | medium | $\beta_1^{medium}$ |
| Neighbourhood | 1 | high | $\beta_1^{high}$ |
| Neighbourhood | 25% | low | $\beta_{25}^{low}$ |
| Neighbourhood | 25% | medium | $\beta_{25}^{medium}$ |
| Neighbourhood | 25% | high | $\beta_{25}^{high}$ |
| Neighbourhood | 50% | low | $\beta_{50}^{low}$ |
| Neighbourhood | 50% | medium | $\beta_{50}^{medium}$ |
| Neighbourhood | 50% | high | $\beta_{50}^{high}$ |
| Neighbourhood | 100% | low | $\beta_{100}^{low}$ |
| Neighbourhood | 100% | medium | $\beta_{100}^{medium}$ |
| Neighbourhood | 100% | high | $\beta_{100}^{high}$ |

Table 3.1: All 24 different experiment settings.

## 3.3 Implementation

The implementation of the experiment consisted of two separate modules: the simulation and the data analysis. While the simulation was written in C++ to leverage the language's high performance, the data analysis part was implemented in Python for its extensive library of analysis and visualisation modules.

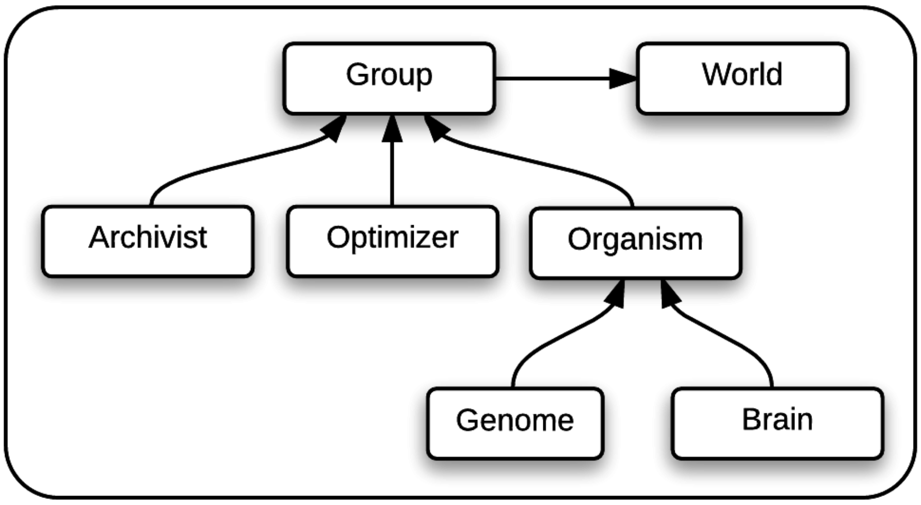


Figure 3.3: An overview over the general structure of the MABE framework [15][14].

The simulation is an extension of the Modular Agent-Based Evolver (MABE) framework developed by Hintze et al [15]. MABE is a framework for developing and running evolutionary algorithm experiments. The framework's first-class support for Markov Brains and its ease of use made it a good choice for this thesis. It provides interfaces and sample implementations thereof for every part of the evolutionary process. An overview of the structure of MABE can be seen in figure 3.3. An *organism* consists of a *brain* that controls the organism's actions and a *genome* that acts as a blueprint for said brain. At the moment of writing this thesis, there are seven different brain types, like the Markov Brain or Long Short-Term Memory Brain, one can choose from. A group of organisms form a *population*. A population is placed inside a *world*, which is essentially equivalent to the setting the experiment is performed in and is responsible for evaluating the organisms. For instance, a world could be a maze organisms have to find a way out of that rewards organisms whenever they find the exit in a given time limit. Once the world finished evaluating the population, the *optimiser* is in charge of generating the population of the next generation by applying the genetic operators on the current set of organisms. The *archivist* is responsible for storing data about the organisms, e.g. objective values or the line of descent. Together, the archivist, the optimiser and the population form a *group*.

For this thesis a custom implementation of a world originally developed in [32] was extended. Instead of evaluating how each organism interacts with clones of itself, every organism is placed in the same room. Since MABE only supports single-objective optimisation in its base framework, the two multi-objective optimisers, i.e. NSGA-II and the neighbourhood algorithm, were implemented as part of this thesis. Additionally, the archivist was slightly changed so that organisms that are archived will have their Transition Probability Matrix (TPM) and Connectivity Matrix (CM) stored as well, which eliminates the need for a second "visualisation" run of the simulation. And lastly, the two multi-objective optimisers also periodically store data about the Pareto fronts like the hypervolume and which organism is part of which front. After performing the simulation, the archived data of every experiment is moved into a sqlite3 database [40] to allow for easier indexing and manipulation.

The PyPhi python package developed by Mayner et al [53] was used for the IIT analysis of the organisms' brains. It acts as a reference implementation of IIT, enabling users to analyse the cause-effect structures and integrated information of networks such as those used in this thesis. Other python libraries used in the analysis of the experiment are pandas [54], which is an open source library containing various data analysis tools, matplotlib [43], a 2D plotting library, and seaborn [89], a high-level data visualisation library based on matplotlib.

# 4 Evaluation

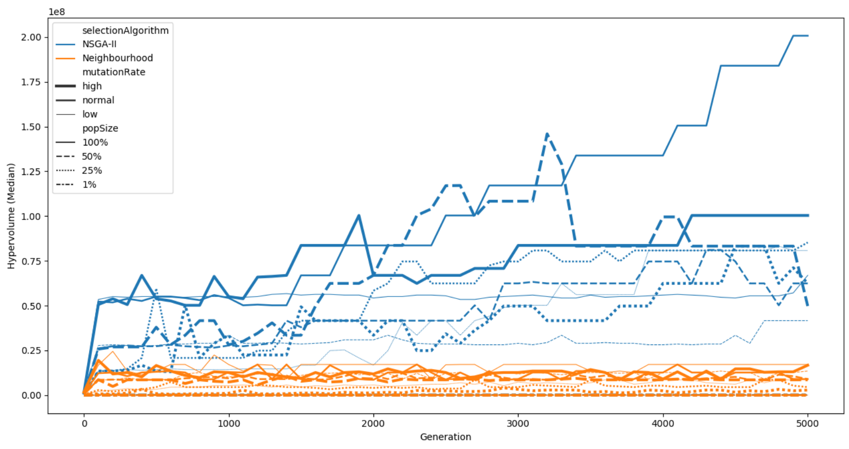
In this chapter we will perform an analysis of the results of the experiments described in the previous chapter. The first section focuses on the multi-objective nature of the simulation by analysing the evolution of the hypervolume and correlations between the objectives. The second section then takes a closer look at the integrated information generated by the produced organisms.

## 4.1 Multi-objective Analysis

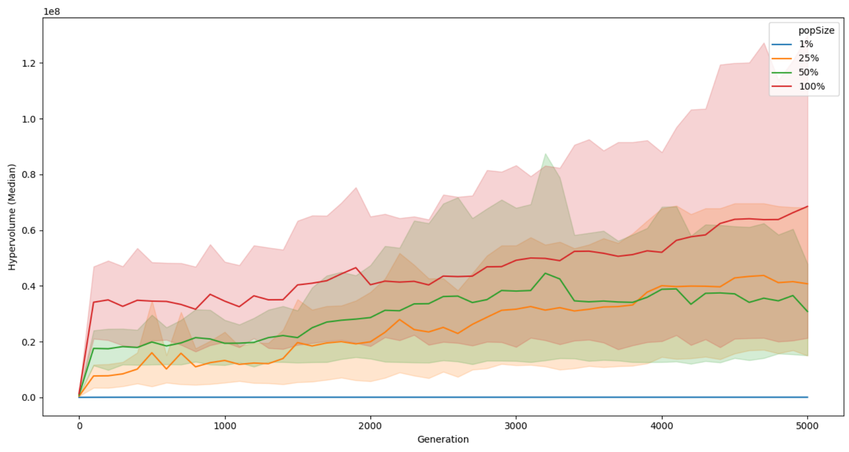
Before diving into the analysis of the organisms' integrated information, we first want to evaluate the convergence and diversity of the population. For this we look at both the hypervolume of each experiment with a focus on the influence of each of the parameters, followed by a front analysis and a correlation analysis of the three objectives.

### 4.1.1 Convergence

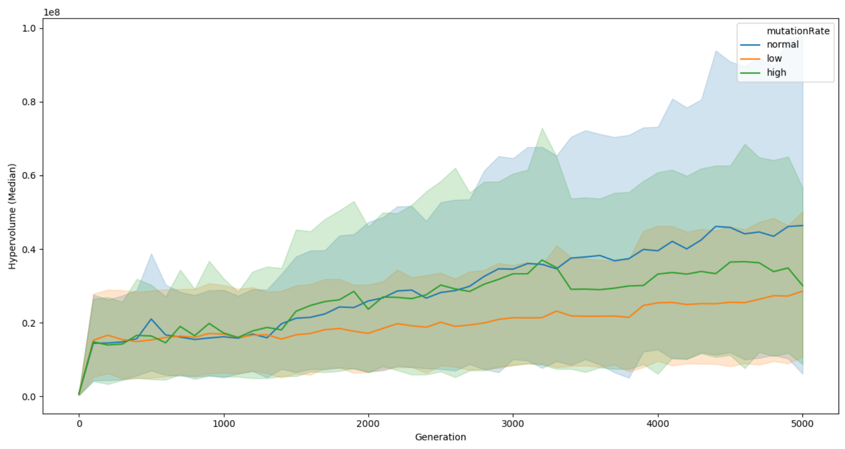
Figure 4.1 shows three different plots, which each visualise the evolution of the population's hypervolume for every experiment. Since each experiment is repeated 31 times, the median of the repeated experiments is used. The first line plot shows a line for each of the 24 different experiments with a different colour, style and thickness depending on the selection algorithm (*selectionAlgorithm*), mutation rate (*mutationRate*) and population Size (*popSize*) parameters, respectively. From this plot it is apparent that the experiments that used the NSGA-II algorithm, shown here in blue, all have a higher median hypervolume than those that applied the Neighbourhood algorithm. The best performing combination of parameters in terms of hypervolume also has a normal mutation rate and a population size of 100%, while the worst combinations all have a population size of 1.



(a) Hypervolume of all 24 experiments.



(b) Hypervolume grouped by population size.



(c) Hypervolume grouped by mutation rate.

Figure 4.1: Evolution of the population's hypervolume

The following two plots focus on the effect of the mutation rate and the population size. Experiments with the same mutation or population size parameters are grouped together, i.e. there are as many lines as there are different parameter values. Each line is the mean of all corresponding values and the area around each line is the Standard Error of the Mean (Standard Error of the Mean (SEM)). One can see that a fully populated simulation ($P_{100}$) also results in the highest hypervolume on average. However, the standard error is also much higher than for less densely populated experiments. The reason for this large span could be that more organisms usually result in more opportunities for collisions and thus a lower hypervolume, while the larger population size also allows for a higher diversity, increasing the hypervolume. Depending on the random seed, this could then result in either a low or a really high diversity and convergence. Population sizes of 25% ($P_{25}$) and 50% ($P_{50}$) seem to perform similarly. Until around generation 4000 $P_{50}$ had a slightly higher average Hypervolume (HV) than $P_{25}$ and even came close to the HV of $P_{100}$. After that point, $P_{50}$ slowly decreased, allowing $P_{50}$ to overtake it. Experiments with a population size of 1 ($P_1$) have the lowest hypervolume by far, since they have to rely on a good starting genome and mutation to guide the lone organism towards the true Pareto front, even though the collision value is already the best it can be (0).

Even though populations with a medium mutation rate ($P_{normal}$), which are displayed in blue, have the highest mean hypervolume, the differences to $P_{low}$ and $P_{high}$ not being that large in combination with the high spread of all three imply that the mutation rate does not have that much of an effect on the overall performance of population. The mean hypervolume of both $P_{normal}$ and $P_{high}$ are on par with each other for most of the generation until $P_{high}$'s performance stagnates at around generation 3200. $P_{low}$ even overtakes $P_{high}$ in the last few generations because of its steady rise in performance. Overall, the mutation rate doesn't have as much influence on the result as the selection algorithm or population size parameter though.

## 4.1.2 Objective Correlation

Table 4.1 shows the pearson correlation coefficient $\rho$ with the corresponding p values between each of the objectives. It shows that there is no clear universal strong correlation between any of the objectives. The relatively high correlation between collisions and movement penalties is due to organisms blocking

| | | $P$ | $P_{25}$ | $P_{50}$ | $P_{100}$ | $P_\alpha$ | $P_\beta$ |
|---|---|---|---|---|---|---|---|
| C \| M | $\rho$ | 0.135 | 0.247 | 0.139 | 0.023 | 0.084 | 0.181 |
| | p | 0.00e+00 | 0.00e+00 | 0.00e+00 | 6.96e-11 | 1.6e-262 | 0.00e+00 |
| C \| G | $\rho$ | 0.000 | 0.020 | 0.018 | 0.002 | 0.009 | 0.069 |
| | p | 8.75e-01 | 3.73e-09 | 1.64e-07 | 5.46e-01 | 2.90e-04 | 8.2e-107 |
| M \| G | $\rho$ | -0.012 | 0.153 | 0.185 | 0.066 | -0.038 | 0.184 |
| | p | 3.25e-10 | 0.00e+00 | 0.00e+00 | 3.96e-82 | 1.59e-53 | 0.00e+00 |

Table 4.1: Pearson correlation $\rho$ and its p-value for each pair of objectives, which are abbreviated here ($C$ = Collisions, $M$ = Movement Penalties, $G$ = Gate Passages). The X | Y syntax stands for the correlation between objectives X and Y. $P_\alpha$ and $P_\beta$ correspond to the identifiers assigned in 3.1, i.e. NSGA-II and Neighbourhood respectively.

one another. Consider a situation where organism $A$ is standing still in a corner or at some other fixed point *pos* on the grid and organism $B$ approaches *pos*. If $B$'s capabilities don't include circumventing other agents, it will try to move forward through $A$, at which point $B$ is stopped in its tracks and is awarded a collision point. As long as the situation stays the same, i.e. no other organisms move into the field of vision of $A$ or $B$, it is probable that the collisions will keep happening. This causes $A$ to stay at the same place for an extended amount of time, triggering a movement penalty. Only if $A$'s functionality is complex enough to escape from such a situation or if other organisms change the scenario will the cycle stop, i.e. both collisions and movement penalties rise indefinitely. The highest correlation between the two can be observed for $P_{25}$, where collisions are less likely to happen, which means the described scenario has more impact on the data.

One would expect collisions and gate passages to have a high correlation, since more gate passages imply more movement, which would imply a higher chance of encountering another organism, which in turn would lead to a higher collision count. However, there is close to no correlation between the two objectives. One reason for this could be the scenario described above. Another explanation could be that there are a lot of organisms that only move around inside one room, which would increase the chance of collisions without increasing gate passages.

### 4.1.3 Organism Distribution

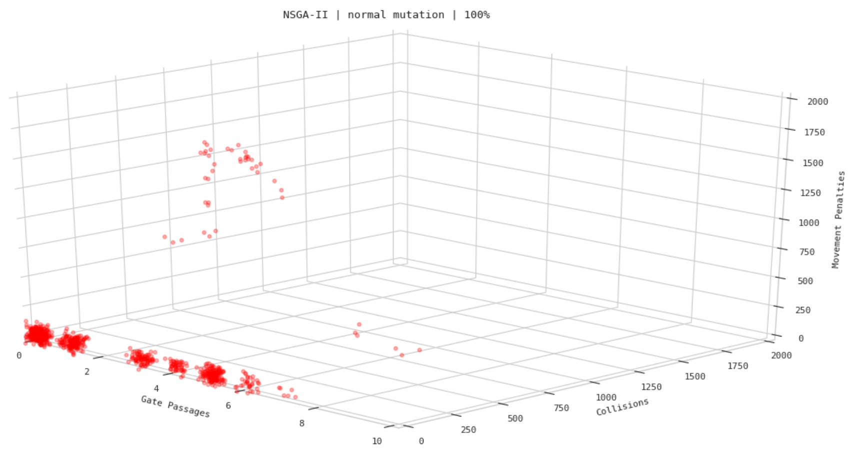
Figure 4.2 shows the objective distribution of $\alpha_{100}^{medium}$ and $\beta_{100}^{medium}$ across all repeats at the end of the simulation, i.e. at the 5000th generation. To avoid overplotting issues only the first Pareto front of each run is visualised. Due to the high density of organisms the points are also both jittered and drawn semi-transparently ($\alpha = 0.3$).

Most of the organisms produced by $\alpha_{100}^{medium}$ perform vastly better than those of $\beta_{100}^{medium}$. While $\beta_{100}^{medium}$'s best performing organisms have at most two gate passages, those from $\alpha_{100}^{medium}$ achieve up to seven. Most organisms in both sets have the same amount of collisions and movement penalties, namely 0, while only differing in their amount of gate passages. However, the population of both experiment settings also include a few organisms that are located off this main cluster. Although both sets of these kinds of organisms are around the same size, those produced by $\alpha_{100}^{medium}$ still perform better than their $\beta_{100}^{medium}$ counterpart on average.

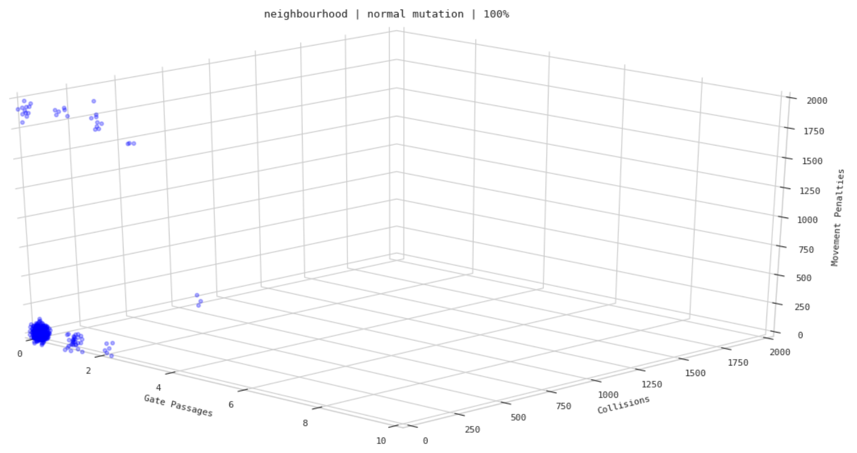
Overall, there is no obvious difference in diversity between the results of the two experiment settings, but a difference in convergence or general performance can be observed. That means $\alpha_{100}^{medium}$'s superior hypervolume (85280360 vs. 4488440) is mostly a result of a better convergence to the Pareto front. One of the reasons for the worse performance of $\beta_{100}^{medium}$ could be the slow diffusion of solutions, which might have slowed down the population's progression. It is possible that a longer running simulation of $\beta_{100}^{medium}$ might generate similarly performing organisms.

## 4.2 Integrated Information

In the following we will analyse the evolution of the IIT measures (see 3.2), the correlations between them and the three objectives and how the experiment parameters affect the overall integrated information. Afterwards, we will take a more detailed look at the brain structure of the best performing organisms of the experiments.



(a) Distribution of organisms for $\alpha_{100}^{normal}$ (see 3.1).



(b) Distribution of organisms for $\beta_{100}^{normal}$ (see 3.1).

Figure 4.2: Scatterplot distributions of two experiments' best organisms, i.e. those that are part of the first pareto front, at generation 5000.

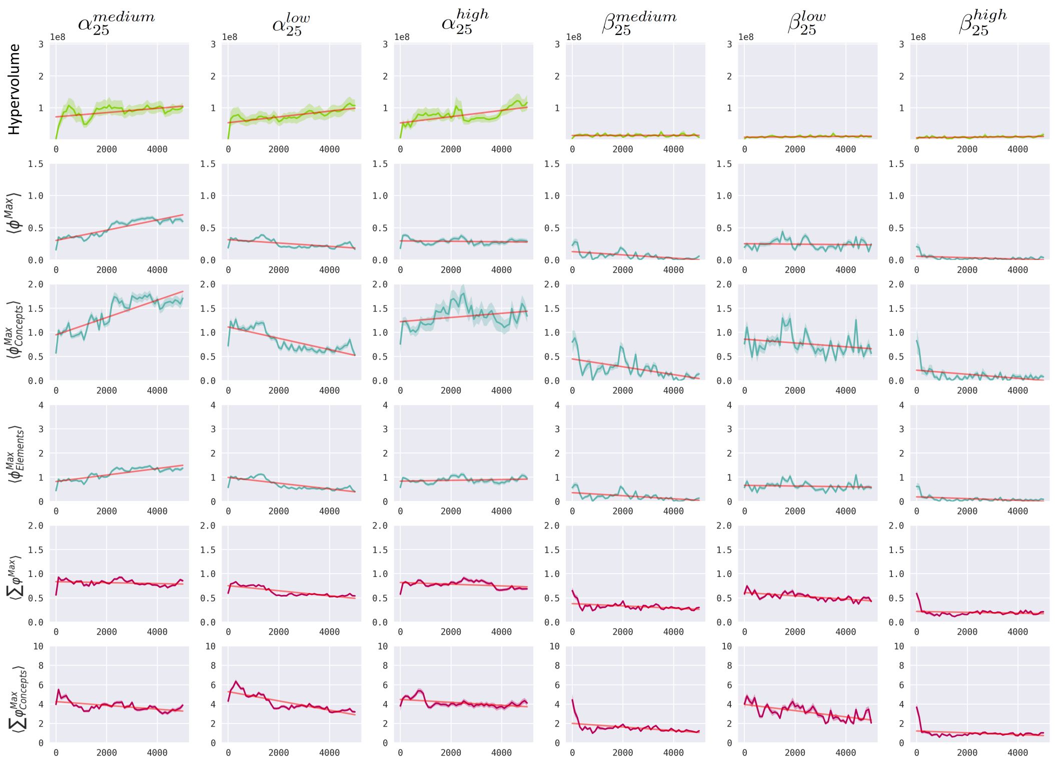


Figure 4.3: Development of IIT measures of $P_{25}$ over time.

### 4.2.1 Development

Figure 4.3 shows the average hypervolume and integrated information of all experiments with a population size of 25% (i.e. $P_{25}$). The area surrounding the average line corresponds to the SEM. Additionally, a linear regression line is displayed in red. To make the plots easier to read, lines of IIT measures that are only concerned with the main complex of the brain are displayed in blue, while the rest are drawn in purple.

When comparing the NSGA-II and Neighbourhood runs, some differences become apparent. The $\Phi^{Max}$ values of $\alpha_{25}$ are all higher on average, with the exception of the low mutation run, in which both $\alpha$ and $\beta$ have a similar performance. The same applies for the other MC centric measures, $\Phi^{Max}_{concepts}$ and $\Phi^{Max}_{elements}$, but is not as noticeable for $\Phi^{Max}_{elements}$. The differences become most notable when contrasting $\alpha^{medium}_{25}$ and $\beta^{medium}_{25}$. $\Phi^{Max}$ and other related measures of the former are rising steadily, while the latter shows the opposite development. The complexity of the whole brain, which is measured in $\sum \phi^{Max}$ and $\sum \phi^{Max}_{concepts}$, of $\alpha$ seems to be higher than that of $\beta$ on the whole.

The mutation rate has a noticeable impact on the integrated information of both $\alpha$ and $\beta$. While the complexity of low mutation rate organisms in $\alpha$ is either stagnant or slowly declining, experiments with high and medium mutation rates appear to produce organisms with an increasingly better integrated main complex over time. For $\beta$, the opposite is true. In both $\beta^{medium}_{25}$ and $\beta^{high}_{25}$, organisms appear to lose their complexity very quickly, while the low mutation rate run is more stable. The reason for this could be that the higher mutation rate combined with the slower spread of good solutions results in well-integrated organisms mutating too often and becoming less integrated before they get a chance to spread.

Figure 4.4 illustrates how the population size influences the evolution of the integrated information. When looking at the $\alpha$ graphs, one can see that a population size of 25% results in a higher level of integrated information on average. Similar to the findings in [32], we assume that this is because organisms in higher density populations don't need a complex brain to generate good results, since they can use other organisms for guidance. The MC complexity of $\alpha^{medium}_{25}$'s population appears to continue rising if the simulation ran longer, which could be explored in a future work.

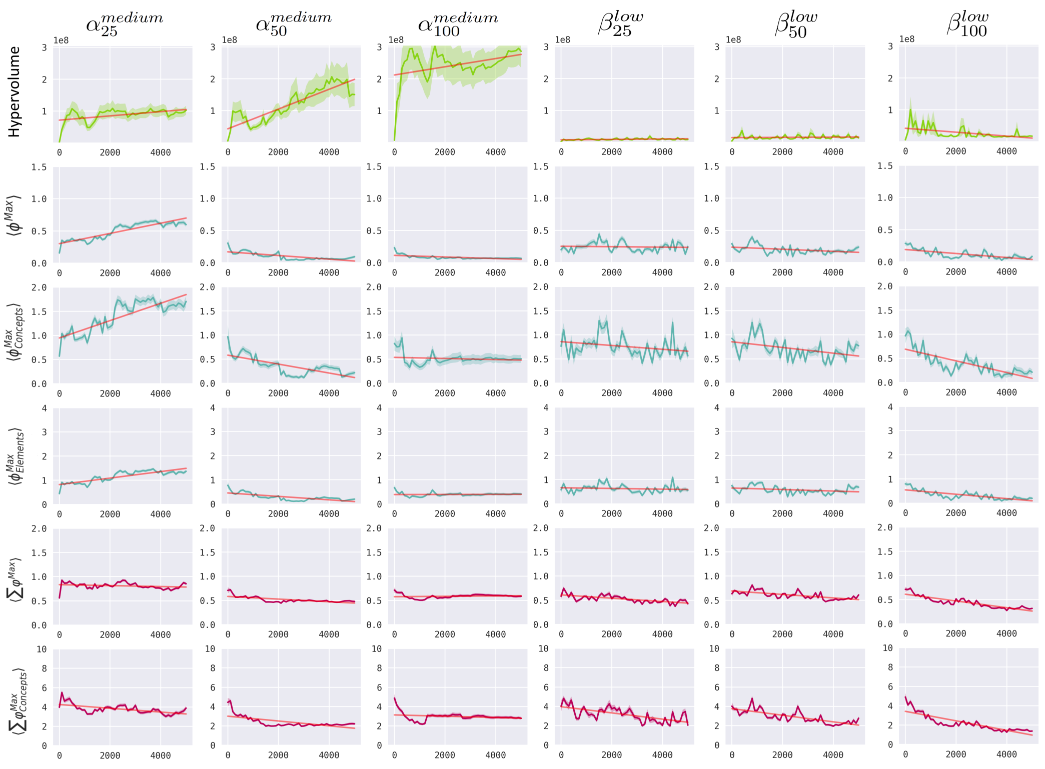


Figure 4.4: Development of IIT measures of three different population sizes for each of the two selection algorithms.

Unlike the $\alpha$ plots in 4.4, the three plots on the right side show the effect of population size on the *low* mutation rate $\beta$ experiments. However, they show a similar trend. While the lower population sizes, i.e. 25% and 50%, display a steady, albeit stagnant, development, $\beta_{100}^{low}$'s integrated information declines over time.

| | | $P$ | $P_{25}$ | $P_{50}$ | $P_{100}$ | $P_\alpha$ | $P_\beta$ |
|---|---|---|---|---|---|---|---|
| C \| $\Phi^{Max}$ | $\rho$ | 0.002 | -0.014 | 0.002 | 0.001 | 0.025 | -0.011 |
| | p | 2.47e-01 | 1.85e-05 | 5.20e-01 | 7.90e-01 | 2.86e-25 | 4.80e-04 |
| M \| $\Phi^{Max}$ | $\rho$ | -0.092 | -0.106 | -0.025 | -0.013 | -0.096 | -0.080 |
| | p | 8.75e-01 | 1.2e-213 | 6.60e-14 | 1.17e-04 | 0.00e+00 | 2.44e-142 |
| G \| $\Phi^{Max}$ | $\rho$ | -0.012 | -0.084 | -0.042 | -0.056 | -0.160 | -0.035 |
| | p | 3.25e-10 | 3.6e-134 | 1.08e-35 | 5.26e-60 | 0.00e+00 | 4.80e-29 |
| C \| $\sum \phi^{Max}$ | $\rho$ | -0.035 | -0.070 | -0.020 | -0.005 | -0.006 | -0.055 |
| | p | 5.76e-77 | 1.55e-94 | 1.47e-09 | 1.71e-01 | 1.59e-02 | 9.53e-70 |
| M \| $\sum \phi^{Max}$ | $\rho$ | -0.279 | -0.224 | -0.063 | -0.067 | -0.268 | -0.319 |
| | p | 0.00e+00 | 0.00e+00 | 8.08e-78 | 2.41e-83 | 0.00e+00 | 0.00e+00 |
| G \| $\sum \phi^{Max}$ | $\rho$ | 0.100 | -0.019 | 0.114 | 0.211 | 0.013 | -0.066 |
| | p | 0.00e+00 | 5.79e-09 | 7.5e-251 | 0.00e+00 | 4.61e-08 | 1.91e-98 |

Table 4.2: Pearson correlation $\rho$ [11] and its p-value for every objective ($C$ = Collisions, $M$ = Movement Penalties, $G$ = Gate Passages) and each of the two main IIT measures.

Table 4.2 shows the Pearson correlation [11] between the optimised objectives and two of the IIT measures, namely $\Phi^{Max}$ and $\sum \phi^{Max}$. When looking at the movement penalty correlations, one can see the trend from above continued here. The larger the population, the less likely it is that a low amount of movement penalties is associated with a high main complex and overall brain complexity, and the other way around. However, the correlation between goal passages and $\sum \phi^{Max}$ appears to become stronger the larger the population is. A reason for this difference in overall brain complexity could be that organisms have to navigate around more obstacles, i.e. other organisms, to perform a gate passage, which might require a well-working decision centre. The complexity of the main complex is not affected in the same way though.

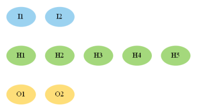
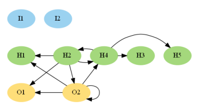
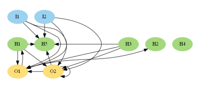
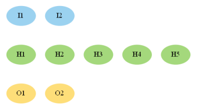
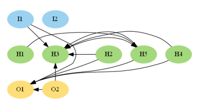
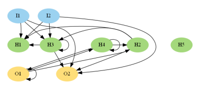
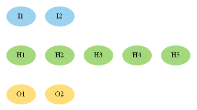
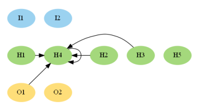
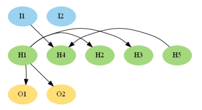
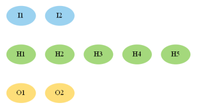
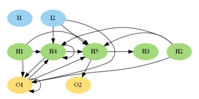
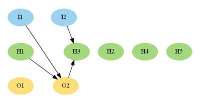
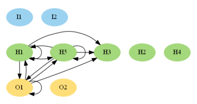
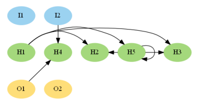
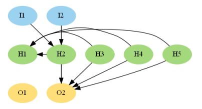
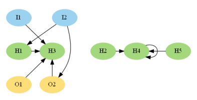
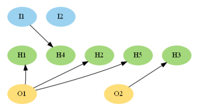
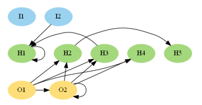
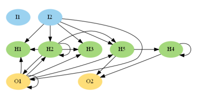
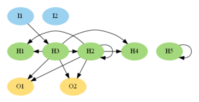
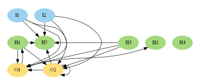
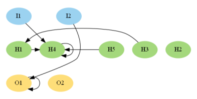
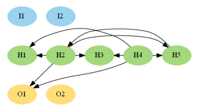
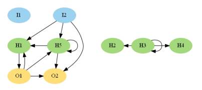
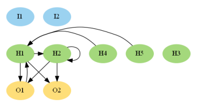
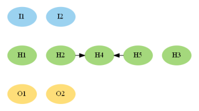
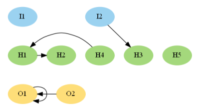
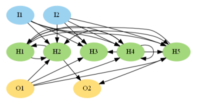
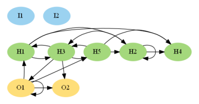
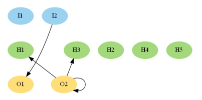
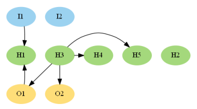
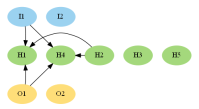
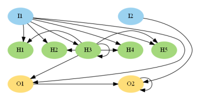
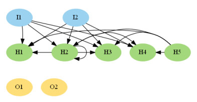
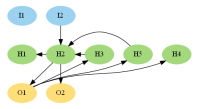
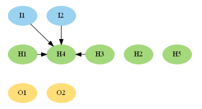
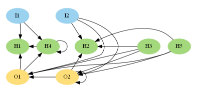
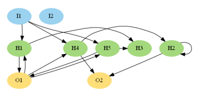
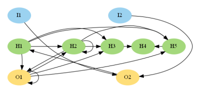
| Experiment | $\max(frequency)$ | $\max(\Phi^{Max})$ | $\max(GatePassages)$ |
|---|---|---|---|
| All |  |  |  |
| $\alpha_1^{medium}$ |  |  |  |
| $\alpha_1^{low}$ |  |  |  |
| $\alpha_1^{high}$ |  |  |  |
| $\alpha_{25}^{medium}$ |  |  |  |
| $\alpha_{25}^{low}$ |  |  |  |
| $\alpha_{25}^{high}$ |  |  |  |
| $\alpha_{50}^{medium}$ |  |  |  |
| $\alpha_{50}^{low}$ |  |  |  |
| $\alpha_{50}^{high}$ |  |  |  |
| $\alpha_{100}^{medium}$ |  |  |  |
| $\alpha_{100}^{low}$ |  |  |  |
| $\alpha_{100}^{high}$ |  |  |  |

Table 4.3: Small multiple of the most common and best performing, according to their $\Phi^{Max}$ and gate passages value, brains of the NSGA-II experiments and overall best brains.

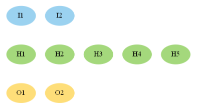
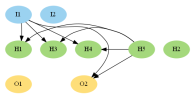
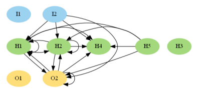
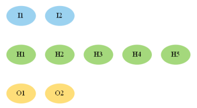
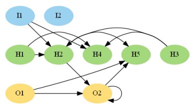
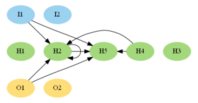
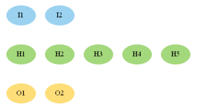
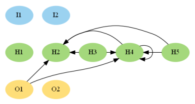
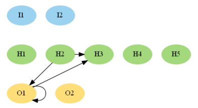
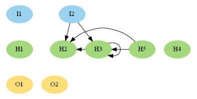
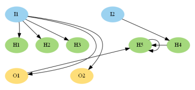
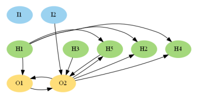
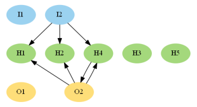
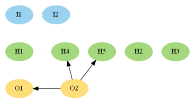
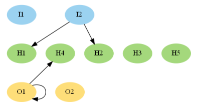
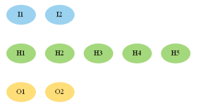
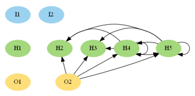
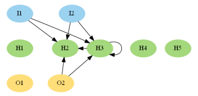
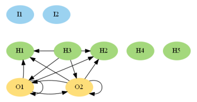
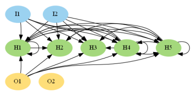
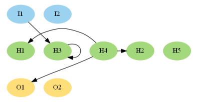
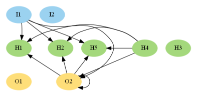
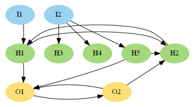
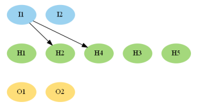
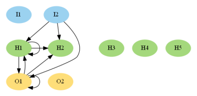
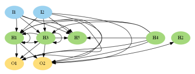
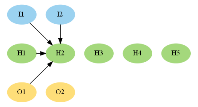
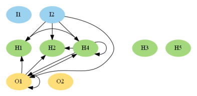
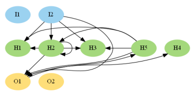
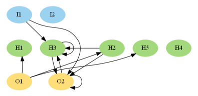
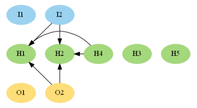
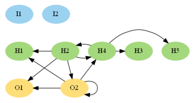
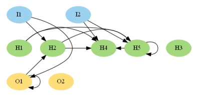
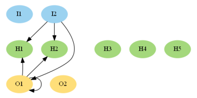
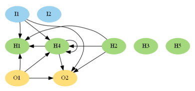
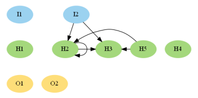
| Experiment | max(*frequency*) | max($\Phi^{Max}$) | max(*GatePassages*) |
|---|---|---|---|
| $\beta_1^{medium}$ |  |  |  |
| $\beta_1^{low}$ |  |  |  |
| $\beta_1^{high}$ |  |  |  |
| $\beta_{25}^{medium}$ |  |  |  |
| $\beta_{25}^{low}$ |  |  |  |
| $\beta_{25}^{high}$ |  |  |  |
| $\beta_{50}^{medium}$ |  |  |  |
| $\beta_{50}^{low}$ |  |  |  |
| $\beta_{50}^{high}$ |  |  |  |
| $\beta_{100}^{medium}$ |  |  |  |
| $\beta_{100}^{low}$ |  |  |  |
| $\beta_{100}^{high}$ |  |  |  |

Table 4.4: Small multiple of the most common and best performing, according to their $\Phi^{Max}$ and gate passages value, brains of the Neighbourhood experiments.

### 4.2.2 Brain Analysis

Tables 4.3 and 4.4 show the most common brains and those with the highest $\Phi^{Max}$ and gate passages values grouped by their associated experiment as a small multiple. Additionally, we chose to include the best performing and most common brains across all experiments ("All"). The graphs representing the brains are divided into three parts or rows: inputs (blue), hidden nodes (green) and outputs (yellow). An edge between two nodes $A$ and $B$ means that the state of $A$ at time step $t$ influences the state of $B$ at $t+1$, i.e. $A_t$ and $B_{t+1}$ are connected through a gate. We chose to use gate passages as the representative of the objectives, because a good gate passages value requires more deliberate action than either of the other two. For instance, an organism can just do nothing for the whole simulation and will have a perfect collision score, but will probably not contain an interesting brain structure.

One of the first things one notices is that the most common brain across all experiments has no edges at all. As expected, these organisms exhibit a poor performance of 0.295 gate passages, 54.311 collisions and 1986.051 movement penalties on average and a mean $\Phi^{Max}$ and $\sum \phi^{Max}$ of 0. They can be mostly found in the $P_1$ experiments, but are also the most common graph structure of $\beta_{25}^{high}$, suggesting that experiments in which mutation plays too much of a role suffer from a loss of complexity and integration over time. The graph with the overall highest integrated information of $\Phi^{Max} = 3.253$ and $\sum \phi^{Max} = 1.108$, displayed in the first row and $max(\Phi^{Max})$ column of 4.3, does not consider its inputs at all, which means the environment has no impact on the organism's output. Consequently, if such an organism would still display a good objective performance, it would probably be the result of mostly luck. This supports the observations from 4.2 that a high integration does not necessarily lead to a good performance. When looking at the graphs that resulted in the highest gate passages values (shown in the last column), one can see the difference more clearly. With a few exceptions, most of these graphs are highly integrated with their input nodes, but generate close to or no integrated information.

When comparing the graphs of $\alpha$ and $\beta$, the differences between the two scenarios become more apparent. $\alpha$'s best gate passages graphs include vastly more edges between nodes than those generated by $\beta$, which could explain the difference in performance and thus hypervolume (see 4.1.3 and 4.2.1). This disparity becomes even more clear when looking at $\alpha^{high}$ and $\beta^{high}$ graphs. For instance, $\alpha_{50}^{high}$'s most common graph is close to fully connected, whereas the neighbourhood equivalent $\beta_{50}^{high}$ includes four nodes without any connection, which are also called orphan nodes. This can be seen in the most common graphs of various other neighbourhood experiments, e.g. $\beta_{100}^{high}$. In fact, the most common graph of $\beta_{25}^{high}$ has no edges at all. Similar to the findings in 4.2.1, we think this is due to the little amount of crossover opportunity for smaller populations in $\beta$, which in combination with the higher mutation rate might result in the loss of brain complexity.

# 5 Conclusion

The detection and quantification of consciousness in machines is an ever-evolving field of research. With more and more products that feature some form of AI becoming part of the everyday life and the simultaneous rise in complexity of such AIs, the ability to identify consciousness in machines is more important than ever. One of the most promising and popular theories in this field is the Integrated Information Theory 3.0 [61], whose mathematical framework aims to make consciousness testable.

To explore and further the research of IIT and machine consciousness in general, we extended the single-objective EA simulation developed by Fischer [32], in which they evaluated the integrated information of a homogeneous swarm of organisms interacting with each other. In order to keep the diversity of the solution set high, we developed a multi-objective GA, for which we identified three objectives: *GatePassages*, which measure how often an organism crosses the goal line of the two-dimensional grid, *Collisions*, which quantify the amount of collisions an organism causes, and *MovementPenalties*, which measures how often an organism performs an unwanted action, e.g. standing still for too long. Additionally, the organisms are evaluated simultaneously as a group, which means they are able to interact with each other in the same simulation run and can thus influence each other's behaviour directly.

To gain a better understanding of the integrated information generated by these organisms and what influence the environment of the organism has on it, we created a set of 24 different experiment parameter combinations out of the mutation rate, population size and selection algorithm choices. For this we implemented a centralised algorithm based on NSGA-II by Deb et al. [26] and a decentralised cellular, i.e. neighbourhood-based, algorithm. The neighbourhood algorithm performs genetic operators such as recombination only inside so-called neighbourhoods, i.e. between organisms whose distance on the grid is smaller than a given distance limit $d_{max}$.

We then evaluated the results of said multi-objective simulation with a focus on the influence of the different parameters. First, we analysed the multi-objective aspects of the results such as the Pareto fronts' convergence and diversity. After an comparison of the evolution of the experiments' hypervolumes, the correlation between the three objectives was examined, followed by an analysis of the front distribution of organisms of two sample experiments. We then evaluated the complexity of the internals of the organisms over time by applying five IIT measures that quantify the integrated information of both the MC and the whole brain and comparing their development over time. Afterwards we investigated the correlation between the simulation's objectives and the IIT measures. Finally, an analysis of the brain structures of the most common and best performing brains was performed.

# 6 Future Work

During the writing of this thesis, we have identified a few problems and research questions that could be explored in a future work.

In this thesis we focused on the development and evolution of a single kind of organism in a static world. Instead of using the same amount of inputs, hidden nodes and outputs for every organism and only changing its wiring, it would be interesting to analyse how organism with completely different internal structures would interact with each other and how this affects the generated integrated information. These different internal structures could also include different gate types.

An interesting idea would be to replace the static grid used here with a more dynamic world, which means organisms would have to be able to adapt to different environments, which could result in more complex and interesting organism structures.

One of the main problems was the high computational complexity of the IIT measures, which severely limited the amount and type of organisms we could analyse. If one were to build a more complex organism structure, a better performing alternative or an approximation of the measures, such as [36] and [5], would have to be used instead.

Another idea that was brought up during development is to use the integrated information of the organisms as an objective in the optimisation process in order to find structures that generate the highest possible integrated information. This would require an interface from the C++ simulation to the python PyPhi module [53] though.

# Bibliography

[1] Enrique Alba and Francisco Chicano. Observations in using parallel and sequential evolutionary algorithms for automatic software testing. *Computers & Operations Research*, 35(10):3161–3183, 2008.

[2] Enrique Alba and Marco Tomassini. Parallelism and evolutionary algorithms. *IEEE transactions on evolutionary computation*, 6(5):443–462, 2002.

[3] Enrique Alba and José M Troya. A survey of parallel distributed genetic algorithms. *Complexity*, 4(4):31–52, 1999.

[4] Larissa Albantakis, Arend Hintze, Christof Koch, Christoph Adami, and Giulio Tononi. Evolution of integrated causal structures in animats exposed to environments of increasing complexity. *PLoS computational biology*, 10(12):e1003966, 2014.

[5] Igor Aleksander and David Gamez. Informational theories of consciousness: a review and extension. In *From brains to systems*, pages 139–147. Springer, 2011.

[6] Javier Apolloni, Guillermo Leguizamón, José García-Nieto, and Enrique Alba. Island based distributed differential evolution: an experimental study on hybrid testbeds. In *Hybrid Intelligent Systems, 2008. HIS'08. Eighth International Conference on*, pages 696–701. IEEE, 2008.

[7] Anne Auger, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler. Theory of the hypervolume indicator: optimal $\mu$-distributions and the choice of the reference point. In *Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms*, pages 87–102. ACM, 2009.

[8] David Balduzzi and Giulio Tononi. Integrated information in discrete dynamical systems: motivation and theoretical framework. *PLoS computational biology*, 4(6):e1000091, 2008.

[9] David Balduzzi and Giulio Tononi. Qualia: the geometry of integrated information. *PLoS computational biology*, 5(8):e1000462, 2009.

[10] Tim Bayne. On the axiomatic foundations of the integrated information theory of consciousness. *Neuroscience of consciousness*, 2018(1):niy007, 2018.

[11] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[12] Susan Blackmore. Consciousness in meme machines. *Journal of Consciousness Studies*, 10(4-5):19–30, 2003.

[13] Clifford Bohm. Brain Markov · Hintzelab/MABE Wiki. `https://github.com/Hintzelab/MABE/wiki/Brain-Markov`. Accessed: 2018-10-04.

[14] Clifford Bohm. Home · Hintzelab/MABE Wiki. `https://github.com/Hintzelab/MABE/wiki`. Accessed: 2018-10-05.

[15] Clifford Bohm and Arend Hintze. Mabe (modular agent based evolver): A framework for digital evolution research. In *Proceedings of the European Conference on Artificial Life 14*, volume 14, pages 76–83. MIT Press, 2017.

[16] Lucas Bradstreet. *The hypervolume indicator for multi-objective optimisation: calculation and use.* University of Western Australia, 2011.

[17] Karl Bringmann and Tobias Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Computational Geometry*, 43(6-7):601–610, 2010.

[18] Rick Cattell and Alice Parker. Challenges for brain emulation: why is building a brain so difficult. *Natural intelligence*, 1(3), 2012.

[19] Michael Cerullo. Integrated information theory a promising but ultimately incomplete theory of consciousness. *Journal of Consciousness Studies*, 18(11-12):45–58, 2011.

[20] James P Cohoon, Shailesh U Hegde, Worthy N Martin, and D Richards. Punctuated equilibria: a parallel genetic algorithm. In *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of*

*Technology, Cambridge, MA*. Hillsdale, NJ: L. Erlhaum Associates, 1987., 1987.

[21] David A Coley. *An introduction to genetic algorithms for scientists and engineers*. World Scientific Publishing Company, 1999.

[22] Hugo De Garis, Chen Shuo, Ben Goertzel, and Lian Ruiting. A world survey of artificial brain projects, part i: Large-scale brain simulations. *Neurocomputing*, 74(1-3):3–29, 2010.

[23] Kenneth A De Jong and Jayshree Sarma. On decentralizing selection algorithms. In *ICGA*, volume 95, pages 17–23, 1995.

[24] Kalyanmoy Deb. Multi-objective optimization. In *Search methodologies*, pages 273–316. Springer, 2005.

[25] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[26] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[27] Grant Dick and Peter A Whigham. Spatially-structured evolutionary algorithms and sharing: Do they mix? In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 457–464. Springer, 2006.

[28] Juan José Durillo, Antonio Jesús Nebro, Francisco Luna, and Enrique Alba. Solving three-objective optimization problems using a new hybrid cellular genetic algorithm. In *International Conference on Parallel Problem Solving from Nature*, pages 661–670. Springer, 2008.

[29] Jeffrey A Edlund, Nicolas Chaumont, Arend Hintze, Christof Koch, Giulio Tononi, and Christoph Adami. Integrated information increases with fitness in the evolution of animats. *PLoS computational biology*, 7(10):e1002236, 2011.

[30] LG Electronics. LG SmartThinQ: Discover LG Smart & Connected Appliances | LG USA. `https://www.lg.com/us/discover/smartthinq/thinq`. Accessed: 2018-10-01.

[31] Michael Emmerich, Nicola Beume, and Boris Naujoks. An emo algorithm using the hypervolume measure as selection criterion. In *International*

*Conference on Evolutionary Multi-Criterion Optimization*, pages 62–76. Springer, 2005.

[32] Dominik Fischer. Recent topics in machine consciousness and the evolution of animats with simulated consciousness and collective behavior. Master's thesis, Otto-von-Guericke University Magdeburg, May 2017.

[33] Mark Fleischer. The measure of pareto optima applications to multi-objective metaheuristics. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 519–533. Springer, 2003.

[34] Lawrence J Fogel, Alvin J Owens, and Michael J Walsh. *Artificial intelligence through simulated evolution.* John Wiley, 1966.

[35] Carlos M Fonseca and Peter J Fleming. Multiobjective genetic algorithms. In *Genetic algorithms for control systems engineering, IEE colloquium on*, pages 6–1. IET, 1993.

[36] David Gamez. Information integration based predictions about the conscious states of a spiking neural network. *Consciousness and cognition*, 19(1):294–310, 2010.

[37] Tushar Goel and Nielen Stander. A study of the convergence characteristics of multiobjective evolutionary algorithms. In *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, page 9233, 2010.

[38] Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, Jun Zhang, Yun Li, Qingfu Zhang, and Jing-Jing Li. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34:286–300, 2015.

[39] Arend Hintze, Jeffrey A Edlund, Randal S Olson, David B Knoester, Jory Schossau, Larissa Albantakis, Ali Tehrani-Saleh, Peter Kvam, Leigh Sheneman, Heather Goldsby, et al. Markov brains: A technical introduction. *arXiv preprint arXiv:1709.05601*, 2017.

[40] D. Richard Hipp, Dan Kennedy, and Joe Mistachkin. SQLite Home Page. `https://www.sqlite.org/index.html`. Accessed: 2018-10-05.

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

[42] Hou-Sheng Huang. Distributed genetic algorithm for optimization of wind farm annual profits. In *Intelligent Systems Applications to Power Systems, 2007. ISAP 2007. International Conference on*, pages 1–6. IEEE, 2007.

[43] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[44] Nikhil J Joshi, Giulio Tononi, and Christof Koch. The minimal complexity of adapting agents increases with fitness. *PLoS computational biology*, 9(7):e1003111, 2013.

[45] Joshua D Knowles, David W Corne, and Mark Fleischer. Bounded archiving using the lebesgue measure. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 4, pages 2490–2497. IEEE, 2003.

[46] Alexey S Kondrashov and James F Crow. Haploidy or diploidy: which is better? *Nature*, 351(6324):314, 1991.

[47] Lukas König, Sanaz Mostaghim, and Hartmut Schmeck. Decentralized evolution of robotic behavior using finite state machines. *International Journal of Intelligent Computing and Cybernetics*, 2(4):695–723, 2009.

[48] Eugene F Krause. *Taxicab geometry: An adventure in non-Euclidean geometry.* Courier Corporation, 1986.

[49] Pedro Larranaga, Cindy M. H. Kuijpers, Roberto H. Murga, Inaki Inza, and Sejla Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2):129–170, 1999.

[50] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3):263–282, 2002.

[51] Dong Hyun Lim, Hoang N Luong, and Chang Wook Ahn. A novel differential evolution incorporated with parallel processing mechanism. In *Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on*, pages 1–4. IEEE, 2010.

[52] Riccardo Manzotti. The computational stance is unfit for consciousness. *International Journal of Machine Consciousness*, 4(02):401–420, 2012.

[53] William GP Mayner, William Marshall, Larissa Albantakis, Graham Findlay, Robert Marchman, and Giulio Tononi. Pyphi: A toolbox for integrated information theory. *arXiv preprint arXiv:1712.09644*, 2017.

[54] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[55] Manfred Milinski and Rolf Heller. Influence of a predator on the optimal foraging behaviour of sticklebacks (gasterosteus aculeatus l.). *Nature*, 275(5681):642–644, 1978.

[56] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[57] Tadahiko Murata, Hisao Ishibuchi, and Mitsuo Gen. Specification of genetic search directions in cellular multi-objective genetic algorithms. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 82–95. Springer, 2001.

[58] Boris Naujoks, Nicola Beume, and Michael Emmerich. Multi-objective optimisation using s-metric selection: Application to three-dimensional solution spaces. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1282–1289. IEEE, 2005.

[59] Antonio J Nebro, Juan J Durillo, Francisco Luna, Bernabé Dorronsoro, and Enrique Alba. Design issues in a multiobjective cellular genetic algorithm. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 126–140. Springer, 2007.

[60] Antonio J Nebro, Juan J Durillo, Francisco Luna, Bernabé Dorronsoro, and Enrique Alba. Mocell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems*, 24(7):726–746, 2009.

[61] Masafumi Oizumi, Larissa Albantakis, and Giulio Tononi. From the phenomenology to the mechanisms of consciousness: integrated information theory 3.0. *PLoS computational biology*, 10(5):e1003588, 2014.

[62] Stephan Olariu and Albert Y Zomaya. Decentralized cellular evolutionary algorithms. In *Handbook of Bioinspired Algorithms and Applications*, pages 121–138. Chapman and Hall/CRC, 2005.

[63] Chrisila B Pettey, Michael R Leuze, and John J Grefenstette. Parallel genetic algorithm. In *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*. Hillsdale, NJ: L. Erlhaum Associates, 1987., 1987.

[64] Henri Pierreval and J-L Paris. Distributed evolutionary algorithms for simulation optimization. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 30(1):15–24, 2000.

[65] TJ Pitcher, AE Magurran, and IJ Winfield. Fish in larger shoals find food faster. *Behavioral Ecology and Sociobiology*, 10(2):149–151, 1982.

[66] Ingo Rechenberg. Evolutionsstrategien. In *Simulationsmethoden in der Medizin und Biologie*, pages 83–114. Springer, 1978.

[67] James A Reggia. The rise of machine consciousness: Studying consciousness with computational models. *Neural Networks*, 44:112–131, 2013.

[68] Nery Riquelme, Christian Von Lücken, and Benjamin Baran. Performance metrics in multi-objective optimization. In *Computing Conference (CLEI), 2015 Latin American*, pages 1–11. IEEE, 2015.

[69] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[70] Robert J Schalkoff. *Artificial neural networks*, volume 1. McGraw-Hill New York, 1997.

[71] Jason R Schott. Fault tolerant design using single and multicriteria genetic algorithm optimization. Technical report, AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH, 1995.

[72] John R Searle. The problem of consciousness. *Consciousness and cognition*, 2(4):310–319, 1993.

[73] Anil Seth. The strength of weak artificial consciousness. *International Journal of Machine Consciousness*, 1(01):71–82, 2009.

[74] Anil K Seth, Eugene Izhikevich, George N Reeke, and Gerald M Edelman. Theories and measures of consciousness: an extended framework. *Proceedings of the National Academy of Sciences*, 103(28):10799–10804, 2006.

[75] Nitasha Soni and Tapas Kumar. Study of various mutation operators in genetic algorithms. *International Journal of Computer Science and Information Technologies*, 5(3):4519–4521, 2014.

[76] William M Spears and Kenneth A De Jong. An analysis of multi-point crossover. In *Foundations of genetic algorithms*, volume 1, pages 301–315. Elsevier, 1991.

[77] Nidamarthi Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[78] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[79] Dirk Sudholt. The benefits of population diversity in evolutionary algorithms: A survey of rigorous runtime analyses. *arXiv preprint arXiv:1801.10087*, 2018.

[80] Ponnuthurai N Suganthan, Nikolaus Hansen, Jing J Liang, Kalyanmoy Deb, Ying-Ping Chen, Anne Auger, and Santosh Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. *KanGAL report*, 2005005:2005, 2005.

[81] Reiko Tanese. Parallel genetic algorithm for a hypercube. In *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*. Hillsdale, NJ: L. Erlhaum Associates, 1987., 1987.

[82] Jing Tang, Meng-Hiot Lim, Yew-Soon Ong, and Meng Joo Er. Study of migration topology in island model parallel hybrid-ga for large scale quadratic assignment problems. In *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*, volume 3, pages 2286–2291. IEEE, 2004.

[83] Marco Tomassini. *Spatially structured evolutionary algorithms: Artificial evolution in space and time*. Springer, 2006.

[84] Giulio Tononi. An information integration theory of consciousness. *BMC neuroscience*, 5(1):42, 2004.

[85] Giulio Tononi. Consciousness as integrated information: a provisional manifesto. *The Biological Bulletin*, 215(3):216–242, 2008.

[86] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[87] David A. Van Veldhuizen and David A. Van Veldhuizen. Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations. Technical report, Evolutionary Computation, 1999.

[88] Lihui Wang, Amos HC Ng, Kalyanmoy Deb, et al. Multi-objective optimisation using evolutionary algorithms: An introduction, 2011.

[89] Michael Waskom, Olga Botvinnik, Drew O'Kane, Paul Hobson, Joel Ostblom, Saulius Lukauskas, David C Gemperline, Tom Augspurger, Yaroslav Halchenko, John B. Cole, Jordi Warmenhoven, Julian de Ruiter, Cameron Pye, Stephan Hoyer, Jake Vanderplas, Santi Villalba, Gero Kunter, Eric Quintero, Pete Bachant, Marcel Martin, Kyle Meyer, Alistair Miles, Yoav Ram, Thomas Brunner, Tal Yarkoni, Mike Lee Williams, Constantine Evans, Clark Fitzgerald, Brian, and Adel Qalieh. mwaskom/seaborn: v0.9.0 (july 2018), July 2018.

[90] Lyndon While. A new analysis of the lebmeasure algorithm for calculating hypervolume. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 326–340. Springer, 2005.

[91] Lyndon While, Philip Hingston, Luigi Barone, and Simon Huband. A faster algorithm for calculating hypervolume. *IEEE transactions on evolutionary computation*, 10(1):29–38, 2006.

[92] Sewall Wright. *The roles of mutation, inbreeding, crossbreeding, and selection in evolution*, volume 1. na, 1932.

[93] Jin Wu and Shapour Azarm. Metrics for quality assessment of a multiobjective design optimization solution set. *Journal of Mechanical Design*, 123(1):18–25, 2001.

[94] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[95] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.

[96] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001.

[97] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer, 1998.

[98] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.

# Declaration of Independence

I hereby declare that this thesis was created by me and me alone using only the stated sources and tools.

Lennart Hoffmann Magdeburg, 2018-11-08