Johan Høye

ArtDev3D: An Artificial Development System

Trondheim, March 2006

Norges teknisk-naturvitenskapelige universitet
Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Computer and Information Science

Master's thesis
Programme of study: Master of Science in Informatics

Supervisor: Pauline Haddow, IDI
Co-Supervisor(s): Diego Federici, IDI
Gunnar Tufte, IDI

NTNU
Innovation and Creativity

Foreword

This master thesis is written by Johan Høye as a part of the Master of Science in Informatics degree at the Department of Computer and Information Science (DIS), Norwegian University of Science and Technology (NTNU). The following assignment description lays the basis for this thesis.

"The assignment is to design and test a novel artificial development system implementing a biologically inspired development process used for the process of mapping the genotype to the phenotype in an evolutionary algorithm."

Abstract

Evolutionary algorithms (EAs) are a class of population-based stochastic search algorithms which have proven themselves to be powerful tools in optimization problems where the search space is complex, contains many local optima, and is so large that an exhaustive search is not possible. An application area where EAs have great potential is in the design of electronic circuits. However, for this type of task such a large representation is typically required for each of the proposed solutions that using an EA approach is not feasible because of the immense computational power this would require. This limitation of EAs is known as the scalability problem: EAs perform well when dealing with problems requiring a small solution representation, but when the required size for these representations increases the EAs quickly become too computationally expensive to be useful.

Numerous approaches for dealing with the scalability problem have been proposed. One of the more promising approaches is inspired by the way nature copes with scaling: the process of an organism growing from a single fertilized cell and into a multi-cellular being, called development. By adapting some of the mechanisms of development to a computer program, the EA can evolve a relatively small genome which when developed i.e. decompressed, using this program will represent a solution. There are, however, some problems regarding this approach. One issue is that biological development is such a complex process that implementing it in all its detail is neither feasible nor desired, meaning a decision regarding which mechanisms to implement and which ones to leave out must be made. Another issue is the increased difficulty to evolve a good solution. This occurs because EAs depend on a gradual refinement of the solution to be effective, but with this approach a small change in the genome may lead to a large change in the corresponding solution. This is because in this approach there is no longer a direct correspondence between the genotype space and the solution space, so that what is adjacent in the genotype space may be far apart in the solution space. This means that even though gradual refinement is achieved in genotype space, the changes in the corresponding solution space may appear to be more or less random

A novel artificial development system, designed and implemented from scratch, is presented in this thesis. A novel system was built because, although a number of other such system already have been implemented, they are all in the experimental stage, and this system is though to be a useful supplement to the existing ones, providing more material to base the understanding of what may be useful in an artificial development system on. An explorative approach was taken where the implemented system was put through a number of tests to investigate its capabilities. First the systems ability to develop a varied set of different shapes was investigated. Secondly, four parameters were tested for their effect on the system's ability to develop good solutions: the initial number of neighbours, the number of chemical types used (both part of a precondition), the number of cell types available to the system, and the degree of symmetry in the target shapes.

The experiments performed showed that the system is able to develop a number of shapes. For the four investigated parameters, indications were found that each has a profound effect on the systems ability to develop a given target.

Acknowledgements

The following people deserves my gratitude for their help during my work on this thesis.

Pauline Haddow for being my supervisor, for giving me valuable advice and for helping me to keep this thesis structured

Diego Federici for fruitful discussions, and an overwhelming amount of creative suggestions

Gunnar Tufte for valuable suggestions during the initial phase of this thesis

Erik Wiborg and **Bjarte Aune Bergstrøm** for correcting my spelling errors and for giving me valuable feedback

Sara and **Jon Noé Høye** for being my supportive and lovely family

Table of Contents

Part I - Introduction and Background

1 Introduction

The way biological development works, starting with only one cell containing the genome which, over time, grows and divides into a mature multicellular organism, is no less than astonishing. It is hypothesised that imitating some aspects of this process in a computer system would make it possible to greatly reduce the needed size for the representation of a solution in an evolutionary algorithm. The need for a rapidly increasing solution size with growing problem complexity is one of the more serious limitation of evolutionary algorithms, inhibiting their usefulness in a number of real-life application areas where they have great potential, such as in the design of digital circuits.

A number of approaches aimed at reducing this problem of scaling have been proposed over the years, like the messy genetic algorithm (mGA) [GOLD89] and the breeder genetic algorithm (bGA) [MUEH93]. The mGA is a variant over the traditional genetic algorithm that pays more attention to the distance between building blocks in the genotype when performing crossover. The bGA is a genetic algorithm inspired by the artificial selection performed by human breeders. This algorithm is suitable for parallel processing as each bGA is responsible for its own population and once in a while they exchange individuals with other running bGAs.

However, lately there has been a growing interest in trying to adapt the way nature copes with scaling – biological development. This approach is so novel that it does not yet have a standardized name. Names like computational development, developmental mapping, embryology and artificial development have been used for this approach in the literature. In this thesis the process will be referred to as artificial development.

The idea behind an artificial development system is as follows: instead of the standard approach of letting the evolutionary algorithm evolve the solution more or less directly, the algorithm evolves a DNA (the genotype) of the solution (the phenotype). The process of development takes the DNA and develops an artificial organism which represents the solution. Fitness of the solution is, therefore, based on the phenotype and not the genotype that is represented within the EA. As the size of the DNA is more or less independent from the complexity of the full-grown organism, this approach has the potential of greatly reducing the size needed for representing a solution in evolutionary algorithms.

In this thesis a novel artificial development system is designed and implemented. An effort is made to design a system that models biological development to some degree, implementing what is thought to be the most relevant mechanisms. Because of the immense complexity of biological development a decision had to be made regarding which mechanisms to include in the system. Implementing biological development in every detail is not desirable both because it would result in an ineffective system which would require tremendous amounts of processing power, and because all mechanisms may not be necessary to fulfil the purpose of such a system. Additionally, even biologists do currently not have a complete understanding of all the intricate details of biological development.

The implemented system is analysed through empirical testing. A decision was made to first investigate the system's ability to develop a varied set of different shapes, and secondly investigate how the changing of certain parameters of the system affects its ability to find good solutions.

The thesis is separated into 4 parts: introduction and background, the development model, experiments, and conclusion and future work.

The introduction and background section begins with this introduction chapter, followed by a discussion regarding the motivation for this thesis in chapter 2. A description of the evolutionary algorithm method used in ArtDev3D, genetic programming, is given in chapter 3. Theoretical background of biological development is presented in chapter 4, and an overview over the field of artificial development is laid out in chapter 5.

The development model section contains three chapters. Chapter 7 gives an overview of the initial system that the author participated in the design of, BioDev. This is followed by a description of the system which is the main focus in this thesis, ArtDev3D, in chapter 8, and in chapter 9 the initial testing of ArtDev3D is performed.

In the experiments section a detailed analysis of the two experiments series performed with ArtDev3D is given. In chapter 10 the effect on the system when changing a chosen set of parameters is investigated, and in chapter 11 a deeper investigation of these effects is performed.

The conclusion and future work section contains two chapters. In chapter 12 the conclusion of this thesis is presented, and in chapter 13 a number of suggestions for future work on ArtDev3D is given.

2 Motivation

My first encounter with evolutionary algorithms (EAs) was back in 1999 when I was surfing the Internet for information on artificial intelligence. Jumping back and forth through more or less interesting pages, I suddenly stumbled into something which caught my attention; evolution as it occurs in nature, adapted to a digital world and used in search problems. First of all, I found it incredibly fascinating that it was possible to move something biological into the digital realm of a computer. Secondly, I just had to learn more about how simulating evolution could be used to perform a search. After reading some pages and viewing a couple of demo's of the algorithm in action, I was past what film-critiques refer to as “the point of no return”; this was something I just had to learn more about!

2.1 Evolutionary Algorithms

EA refers to a class of population-based stochastic search algorithms that are developed from ideas and principles of natural evolution. They include evolutionary strategies [RECH73], evolutionary programming [FOGE66], genetic programming [KOZA92], and genetic algorithms [HOLL75]. Each variant has it's own characteristics and areas of use, but for this chapter it will be sufficient to look at them collectively as a set of search algorithms with more or less equal qualities.

2.1.1 Application Areas

The area where EAs have proven themselves to be most powerful is in optimizing problems where the search space is complex, contains many local optima, and is so large that an exhaustive search is not possible. A classical example of such a problem is the Traveling Salesman Problem, where the optimal (shortest) route between a list of cities is sought. The possible solutions to this problem increases with factorial speed, so as the number of cities to travel increases, traditional search algorithms will have a hard time finding the solution within reasonable time. EAs, on the other hand, is capable of finding a good approximation to the optimal solution relatively quick. While this particular problem perhaps is mainly interesting from a theoretical point of view, EAs also have many various real-world application areas. These include design of artificial neural networks [KOZA91][JONE93], creation of music [SPEC95][JOHA98], evolution of art [SIMS94][ROOK96], and the use of EAs as automated invention machines [KOZA99].

2.1.2 Problems

Given this wide spectre of uses for EAs, it would seem that this set of search algorithms is both versatile and powerful. This is to some extent true, but unfortunately this set of algorithms also have their shortcomings. First of all, setting all the parameters correct is often difficult as there is no universal recipe for how to decide the optimal settings; the values depend greatly on the chosen application area. This is, however, not a major obstacle as “good enough” parameters can be found relatively quick using a trial-and-error approach. More serious are the problems of premature convergence, stagnation, loss of diversity, lack of reliability, lack of efficiency and the fact that EAs do not scale well.

Premature convergence, stagnation and loss of diversity are different aspects of the same problem; the algorithm looses search capability as it progresses. Loss of diversity means that the difference between individuals in the population decreases over time. This is desirable to avoid because it focuses search along a certain path, a path that may well lead in the wrong direction. Loss of diversity may in turn lead to premature convergence and stagnation. The former happens when the search gets stuck in a local optima, the latter when the fitness in the population ceases to increase.

The lack of reliability is closely related to the three problems just described. EAs are not guaranteed to find the optimal solution to a problem, although they are able to find a good approximation to the solution most of the time. Whether this is a problem, depends on the area of application. In many areas, finding the optimal solution is not a necessity, so an approximation will be sufficient.

Because EAs are computationally expensive some efficiency, in term of search powers, has to be sacrificed in order to be able to run them within reasonable time. The efficiency of the algorithm also depends on the representation chosen, where knowledge of the search-space may allow for a biased representation, making the search more effective.

Despite these problems, EAs perform reasonably well on problems where the solution can be expressed in a short, compact form. However, when the size and complexity of the solution increases, the computational requirements of the algorithm increases at a much faster rate, making it impractical to use. This is known as the scalability problem, and overcoming this obstacle is the main driving force behind this thesis. The scalability problem will now be described in greater detail along with a discussion of the work done so far to overcome this problem.

2.2 The Scalability Problem

Scalability is a term used to encompass how good an EA is at solving problems as their respective solutions grow larger and more complex. Because many real-world problems where EAs are applicable has a solution which cannot be represented in a short manner, it is a desired property of the EA that it is able to scale in order to handle also these problems. Unfortunately, this is currently not the case.

2.2.1 The Need For Scaling

EAs are computationally expensive: the need for several hundred individuals going through thousands of evolutionary steps, requires both a big memory to hold the individuals, and a fast processor to drive the evolutionary process. Additionally, both evaluating the fitness for each individual and expressing the solutions may also be computationally expensive. This makes the use of EAs impractical when the solution to the problem in question reaches a certain complexity.

This scalability problem is a major obstacle in such areas as digital circuit design [VASS00], artificial neural network [CORB03], and robot body design [HORN03]. These are areas where

EAs bear promise of good performance, and a lot of effort has been put in to find ways to overcome this problem. Unfortunately, the effort so far has not led to any satisfactory solution.

2.2.2 Approaches Towards Overcoming the Scalability Limitation

A number of suggestions regarding how to overcome the limitation the scalability problem puts on EAs have been put forth. A selection of the most successful approaches will now be presented.

Jim Tørresen suggests, in [TORR98], a scheme called *increased complexity evolution*, which is based on a divide-and-conquer methodology. This scheme is used to evolve complex systems in the field of evolvable hardware. The scalability problem is reduces in that the system in question is based on the evolution of several sub-systems. This approach was later picked up again and extended to an adaptive divide-and-conquer methodology by Purshouse and Flemming in [PURS03]. Their approach decomposes a global problem into several independent sub-problems where this is possible. These sub-problems may then be solved in parallel. The results from the experiments performed are promising, but the automatic identification of independent sub-problems is tricky, and the methodology infers some overhead.

Another approach, suggested by [GOLD89], is called *messy genetic algorithm* (mGA). This is a variant over the traditional genetic algorithm where the linkage between building-blocks is taken much more seriously. Linkage is a term used for the probability that two building blocks will remain together after crossover. The closer two building blocks are, the tighter they are said to be linked. The algorithm first chooses a set of suitable building blocks, and then these building blocks are labelled so that their ordering can be evolved. A major drawback of mGA is that the initialization procedure of finding suitable building blocks is a computationally intensive process. A modified version of the mGA, called *fast messy Genetic Algorithm* (fmGA), was later developed to remedy this problem. The fmGA's use of building blocks is supposed to make it more scalable, and results showing it is able to solve difficult problems fast and reliable have been found [GOLD93]. Unfortunately, this algorithm is difficult to use, which may explain why it is not more widely used. It requires fine tuning of parameters, which can often be a complicated task [HARI97].

Some of the ideas from fmGA is also used in what is called the *linkage learning genetic algorithm* (llGA), introducing introns (non-coding building blocks), and a circular representation of the genome where the building-block are allowed to move freely around [HARI97]. Moreover, the llGA doesn't identify the building block beforehand, as do the fmGA. The llGA performs well (solvable within linear time) on problems where the building blocks are exponentially scaled[1]. When the building blocks are uniformly scaled[2], however, the llGA needs a population growing exponentially with the problem size.

The *breeder genetic algorithm* (bGA) is yet another algorithm dealing with the scalability problem. It was put forth by [MUEH93] in 1993, and can be seen as a recombination between evolution strategies and genetic algorithm. The bGA is inspired by the artificial selection performed by human breeders. The algorithm can run distributed as a collection of bGAs, and

1 Exponentially scaled problems have sub problems of different importance.
2 Uniformly scaled problems have sub problems of equal importance.

then each bGA is viewed as a virtual breeder with the sole responsibility for its own population. Once in a while, the virtual breeders exchange individuals to promote global progress. The bGA is easy to use (only the population size has to be decided by hand), and is a robust global optimization method which has been successfully applied to a number of real world applications.

Others, like [THIE99], have made an effort to overcome the scalability problem by doing some minor changes to the traditional GA. In this case, three techniques were used: *elitist recombination*, *niching* and *restricted mating*. Unfortunately, this approach did not achieve much success.

A number of researchers, like [BENT99], [HADD01], [MILL03] and [FEDE05], have lately begun looking to nature for inspiration. The way nature is able to shrink the information required for a one-to-one mapping of a multicellular organism, like a human, into a relatively small and information poor DNA is no less than astonishing. If it is possible to model this in a computer, it would make it possible to greatly shrink the size of the genotype, thereby reducing the impact of the scaling problem. The process of "decompressing" the DNA into a full-grown organism is called development, and is that which will be discussed next.

2.3 Development – The Way Nature Copes With Scaling

The process in which a fertilized cell grows into a multicellular organism is called development. Several factors guide this process, including the organisms DNA (genome) and its environment. Development as it occurs in nature includes three features which would be desirable to achieve in an artificial development system: it is scalable, flexible and robust.

2.3.1 Scalability

As already stated, development is natures way of coping with scaling. Consider the differences in both size and complexity between a mature human and a mature mouse. The differences are tremendous, but when looking at their respective genome, the difference in size is relatively small. This is possible because the genome do not contain explicit information about every details of the body, instead it serves as a building plan, describing how the organism is to be build. Or, as the vice president for medical research at Howard Hughes Medical Institute puts it, "complexity does not come from the number of genes but from the way in which they are used" [RUBI00].

2.3.2 Flexibility

Nature exhibits great flexibility in expressing a wide array of wildly different organisms, all developed from their respective genome. Achieving this feature in an AD system could be useful because it would make the system generally applicable. Pursuing this feature is, however, currently not a goal for artificial development systems.

2.3.3 Robustness

Living organisms display an incredible robustness towards accidental changes to their body. This is because destroyed cells may be regenerated – replaced by new cells. A typical example of this is the salamanders ability to regenerate a lost limb. Humans also possess this ability, but in a more limited way. This feature would be very valuable to achieve in an AD system because it would enable a device created through the AD system to be able to repair faulty components, without the need for external technical assistance [FEDE05].

2.4 Motivating Work

To create an AD system implementing all details of biological development is not feasible. This is both because of the extreme complexity of biological development and because all the details regarding it are currently not known, even to biologists. Hence, some details need to be left out. Exactly how much detail to implement and how much to leave out is still a debated issue amongst computational development researchers. Some, like [DELL95], [EGGE97] and [BENT99] have made an effort to create biologically plausible models, while others, like [LUKE96], [DITT98], [HADD01] and [MILL03], are merely inspired by biology. Both approaches have their advantages and disadvantages. By modelling biology closely, more of the desired features may be included in the system. However, this results in a more complex development process, requiring more computational power. By only implementing the mechanisms who are thought to play a key role in development, the computational requirements is lowered, resulting in a more efficient system.

The system designed and implemented in this thesis, ArtDev3D, is greatly inspired by the work of Miller [MILL03] and Federici [FEDE04]. An effort was made to create a system that would be, to some extent, biological close. However, it was deemed more important to achieve a certain amount of efficiency, and to try to extract only the most essential mechanisms of biological development, than to model it in great detail. Hence, some biological elements had to be sacrificed for efficiency. Using this approach, it was hoped that the resulting system would provide a both minimalistic and effective alternative to the current artificial development systems.

3 Genetic Programming

As mentioned earlier – see chapter 2.1, there exists a number of evolutionary algorithms. In this thesis only genetic programming (GP) is used. Therefore, a description of GP will now by given, while the other variants will not be discussed any further.

The major components of GP will be described first, followed by an overview of the mechanisms needed for evolution. Finally, the basic version of the GP algorithm will be presented.

This chapter is in large part based on the book "Genetic Programming – An Introduction" by Banzhaf, Nordin, Keller and Francone [BANZ98].

3.1 The Components

A GP has a population consisting of a number of individuals. Each individual has both a genotype and a phenotype. The genotype is the way the individual is represented inside the GP, while the phenotype is the version of the individual which is tested by the fitness function.

The most common way to represent the genotype in GP, is in a tree structure. The tree structure consists of nodes along with lines connecting the nodes to each other. The nodes are categorized into two groups: functions and terminals. Functions are nodes with one or more nodes as input while terminals are nodes without any input. See figure 3.1 for an example of a genotype. This genotype is a tree structure representing the expression "3 + (2 * 7)". Both the functions + (plus) and * (multiply) take two arguments as input. The multiply function has the two terminals "2" and "7" as input, while the plus function has the terminal "3" and the result from the multiply function as inputs.

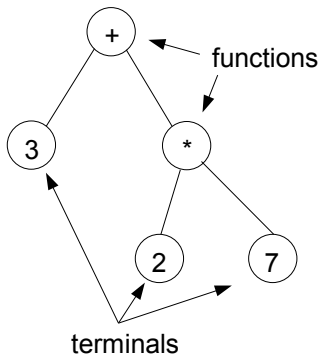


*Figure 3.1: Tree structure representing the expression "3 + (2 * 7)"*

The allowed values and functions for the nodes must be decided before running the GP, and care must be taken when choosing these values and functions so that the solution will be expressible through them. Two sets must be decided, one for the functions and one for the terminals. The function set is comprised of the statements, operators and functions available to the GP system. The terminal set is comprised of the inputs to the GP program and the supplied constants.

3.2 The Mechanisms of Evolution

The following is a description of the mechanisms used by the GP both to achieve evolution, and to guide the evolution in the correct direction. These mechanisms include three genetic operators (search operators): reproduction, crossover and mutation. Additionally, one mechanism for calculating fitness and one for selecting individuals for mating, is needed.

3.2.1 Reproduction

Reproduction is a straightforward operator; select an individual, create a copy of it and place the copy in the population for the next generation.

3.2.2 Crossover

Crossover is considered to be the most important, and effective search operator in a GP system. When performing crossover, one node in each of the two parents are chosen at random. These nodes, along with their corresponding sub tree of nodes, are then exchanged between the parents, creating two offspring. See figure 3.2 for an example of this. The root node of the sub tree enclosed by the dotted circle in each of the two parents are chosen at random. Both of the sub trees are cut out from their respective tree structure, and attached to the other tree structure at the place where the other sub tree was cut out. This results in two offspring, both containing parts of both parents.

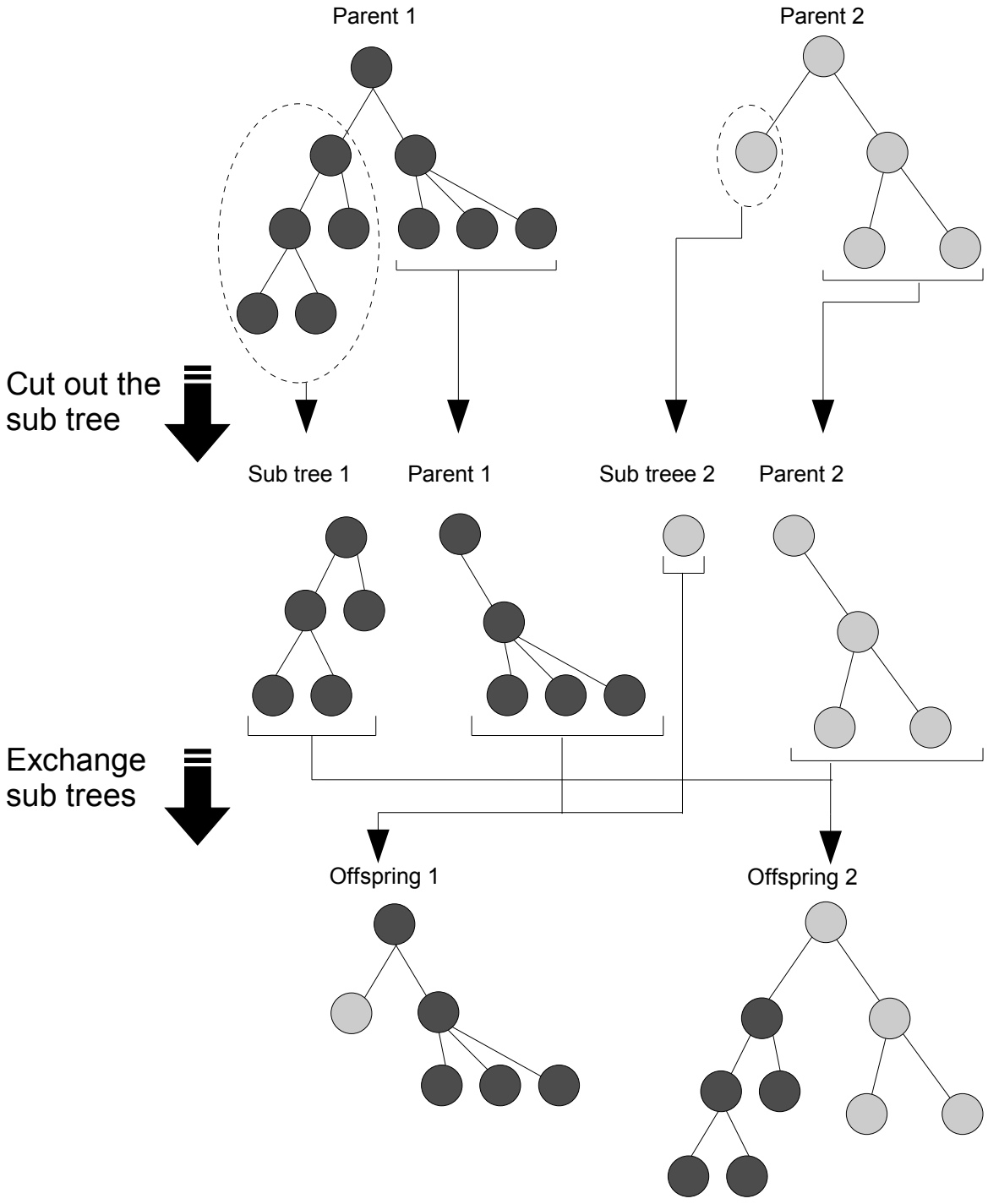


Figure 3.2: Crossover between two individuals creates two offspring, each containing a subset of the nodes from both of the parents

Why is the crossover operator considered to be so important? In [KOZA92], Koza argues that a GP population contains what is called *building blocks*. A building block can be any sub tree present in the population. Individuals containing good building blocks will achieve higher

fitness scores, and hence be more likely to be selected for crossover or reproduction. This increases the possibility for good building blocks to multiply and spread across the population. The hypothesis is as follows: the good building blocks combine into even larger and better building blocks, thereby further increasing the fitness in the population. This hypothesis is based on the schema theorem [HOLL75], and follows the same line of reasoning as the building block hypothesis for genetic algorithms [GOLD89b].

3.2.3 Mutation

The mutation operator is used on one individual at a time. A random node in the individual is selected, and this node, along with its sub tree, is replaced by a randomly generated sub tree. An example of this is given in figure 3.3.

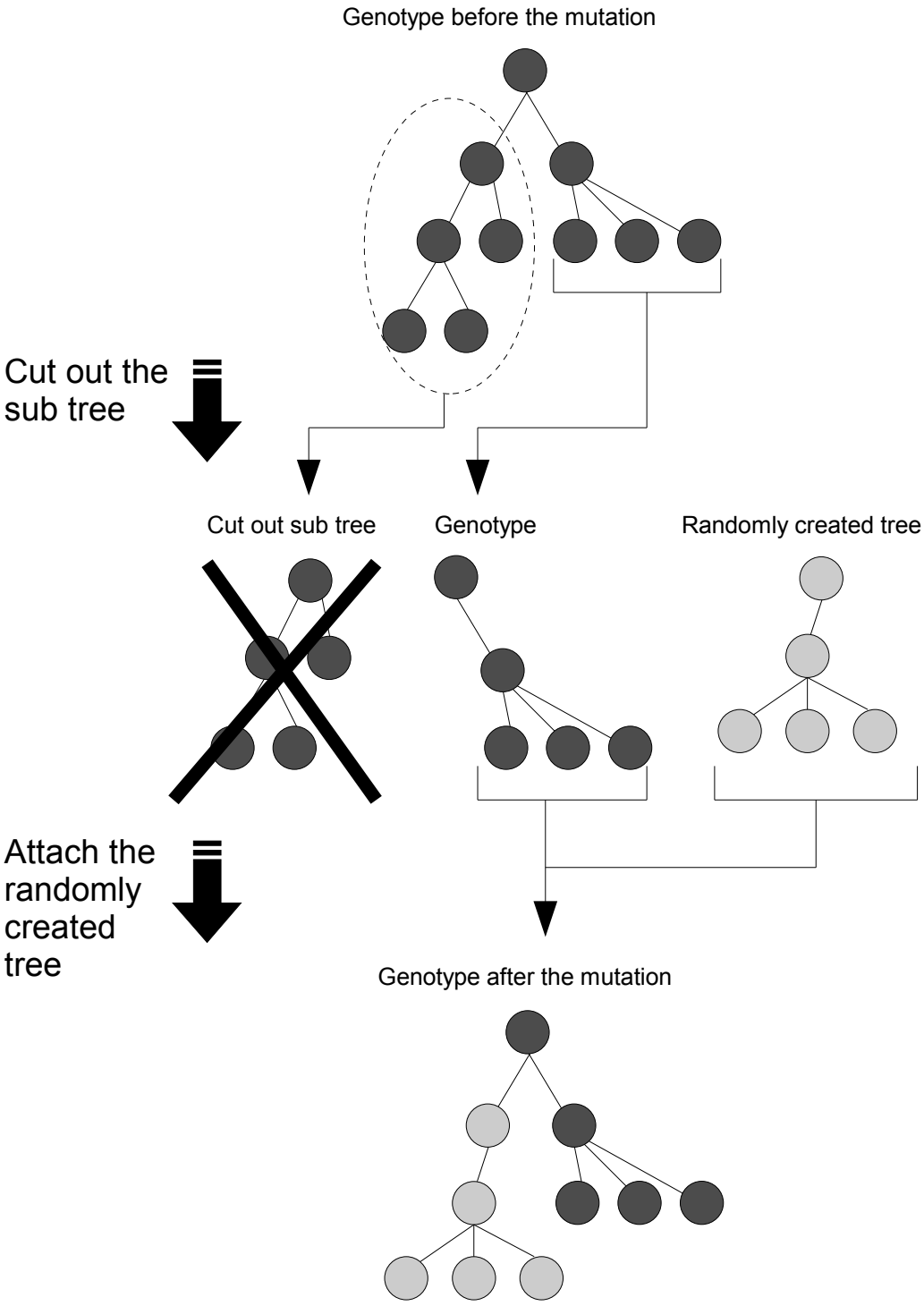


Figure 3.3: Mutation replaces a sub tree within the genotype with a randomly created tree.

Usually, each individual resulting from a crossover or reproduction has a certain chance of being mutated. The probability of mutation is typically set to a low value.

3.2.4 Fitness Function

Fitness is the measure of how well an individual performs when tested against the given goal. In other words, it is a measure of the quality of the individual. The higher value, the more fit the individual is. Other fitness schemes are also possible, like standardized fitness and normalized fitness. With standardized fitness, a value of zero denotes the most fit individual. The advantage of this scheme is that the best fitness is always the same value. The normalized fitness is always in the range zero to one, where zero is the worst fitness and a value of one is the best.

The fitness function evaluates and assigns the fitness value for each of the individuals. To be able to guide evolution in a good way, the fitness function should be designed so that it gives graded and continuous feedback to the GP.

Examples of fitness functions includes:

- The number of matching pixels in an image matching application
- The deviation between prediction and reality in a prediction application
- The money won by a GP-controlled agent in a betting game

3.2.5 Selection Mechanism

Selecting two individuals for crossover or reproduction is the job of the selection mechanism. It uses the fitness assigned to the individuals by the fitness function when performing the selection. There exists a variety of selections algorithms, each having a different effect on the way evolution is guided.

Examples of selection algorithms includes:

- Fitness-proportional selection – An individual is given a probability, proportional to its fitness, of passing an offspring into the next generation. In order to calculate this probability, the fitness of all the individuals in the population must first be evaluated.
- Ranking selection – The selection probability for an individual is a function of their rank in the population. The rank of an individual is decided by its fitness; the more fit, the higher rank.
- Tournament selection – A subset of the population, called a tournament group, is selected at random, and a selective competition between the individuals in this group takes place. Selection pressure is controlled through the size of the tournament group; small groups give low pressure, while bigger groups give higher pressure.

3.3 The Algorithm

The evolution process may be run in two modes: either with distinct generations, or without generations. The former is called generational evolution, while the latter is called steady-state

evolution. Generational evolution is the technique traditionally used in GP, and is also the one used in this thesis, hence the steady-state evolution will not be discussed further.

The basic generational GP algorithm is as follows:

1. Generate a population of randomly created individuals
2. Evaluate each individual and assign a fitness value based on similarity with the target
3. Repeat the following until the new population is filled up
    - Select two individuals from the current population, based on their fitness
    - With a certain probability, perform crossover between the two individuals, creating two offspring
    - With a certain probability, perform mutation on the first offspring
    - With a certain probability, perform mutation on the second offspring
    - Insert the two offsprings into the new population
4. Check if the termination criterion is fulfilled. If it is, continue, if not, repeat steps two and three
5. Present the individual in the last population having the best fitness as the solution

4 Biological Development

All living organisms are composed of one or more cells. Multicellular organisms, like vertebrates, start their existence as a single cell, the fertilized egg. The process of this single cell, the zygote, growing into a full-grown multicellular organism, is called development. During this process the developing organism undergoes a series of progressive changes. These involve cells growing, reproducing and dying. The changes continue even after the organism has reached its full-grown stage, as they are vital for maintenance of the organism.

Three kinds of development are distinguished in biology:

- Chemical: atoms -> molecules
- Cellular: cells -> tissues -> organs -> organ systems - > organisms
- Ecological: organism -> population -> community -> ecosystem -> biosphere

The relevant one for this thesis is cellular development, which will now be described in greater detail.

This chapter is in large part based on the book “Biology” by Raven, Johnson, Losos and Singer [RAVE05].

4.1 Overview

The cells in an organism are not just clustered together in a random, unstructured blob. In vertebrates, cells are organized into tissues, which are groups of cells similar in structure and function. Tissues are, in turn, organized into organs. Organs are body structures composed of several different tissues that form a structural and functional unit. An example of such an organ is the heart. Organs are in turn grouped into organ systems, groups of organs that cooperate to perform the major activities of the body. An example of this is the digestive system, which consists of the stomach and the gallbladder, among others.

Biological macromolecules are the basic chemical building blocks for all organisms. They are large, complex assemblies of functional groups – atoms bonded together in a constellation with definite chemical properties.

Macromolecules are traditionally grouped into four major categories: nucleic acids, proteins, lipids and carbohydrates.

- Nucleic acids are DNA and RNA. DNA encodes genes, and RNA is needed for gene expression. Both will be described in greater detail in chapter 4.1.1.
- Proteins are complex and versatile molecules able to carry out a wide array of functions. Proteins will be described in chapter 4.1.2.

- Lipids are a diverse group of macromolecules. Their functions ranges from energy storage to chemical messengers.

- Carbohydrates are used as energy storage, cell walls and structural support.

The discussion on macromolecules in this thesis will be limited to proteins, DNA and, to some extent, RNA.

The structure and function of the DNA will be described next. This is followed by a section on the protein and finally a section on the cell.

4.1.1 The DNA

Nucleic acids are the information storage and expression devices of the cell. There are two varieties of nucleic acids: *deoxyribonucleic acid* (DNA) and *ribonucleic acid* (RNA). Unique among macromolecules, nucleic acids are able to serve as templates to produce precise copies of themselves. This means the information that specifies what an organism is can be copied and passed down to its descendants. Because of this, DNA is often referred to as the hereditary material.

DNA is a long polymer of repeating subunits called nucleotides. Each nucleotide consists of three components: a five-carbon sugar (deoxyribose), a phosphate group (-PO_4), and an organic nitrogenous base – see figure 4.1. Two types of organic bases occur in nucleotides: purines and pyrimidines. Purines are large, double ring molecules, which in DNA exists in two variants: adenine (A) and guanine (G). Pyrimidines are smaller, single-ring molecules. They also come in two variants in the DNA: cytosine (C) and thymine (T). The information held by the DNA is encoded as sequences of these 4 bases, somewhat similar to the way the letters on a page encode information.

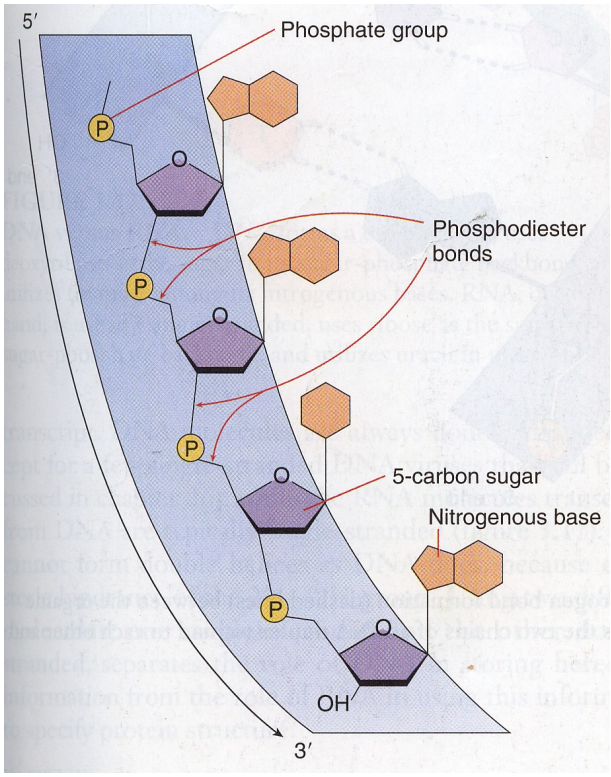


Figure 4.1: The chemical components of the DNA molecule [RAVE05]

The DNA molecule has a characteristic double helix shape – see figure 4.2. Two chains of what is called DNA polymers wind around each other like the outside and inside rails of a spiral staircase. Each step of the DNA's helical staircase is a base-pair, one from each of the two chains, held together by a hydrogen bond. The rules for this base-pairing are rigid: A can only pair with T, and C can only pair with G.

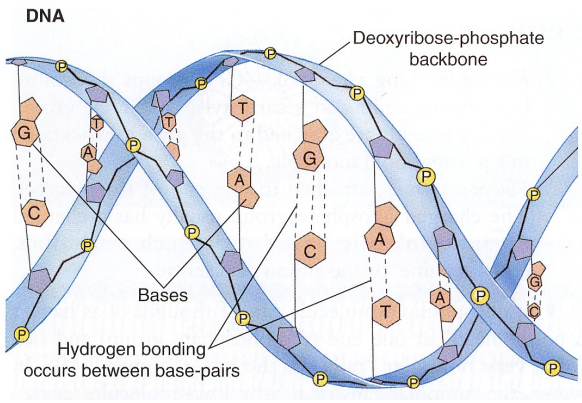


Figure 4.2: The structure of the DNA molecule [RAVE05]

RNA is used to read the DNA-encoded information and direct the synthesis of proteins in cells. RNA is similar to DNA in structure – see figure 4.3. There are three types of RNA, each having a different function: rRNA (ribosomal RNA), mRNA (messenger RNA) and tRNA (transfer RNA).

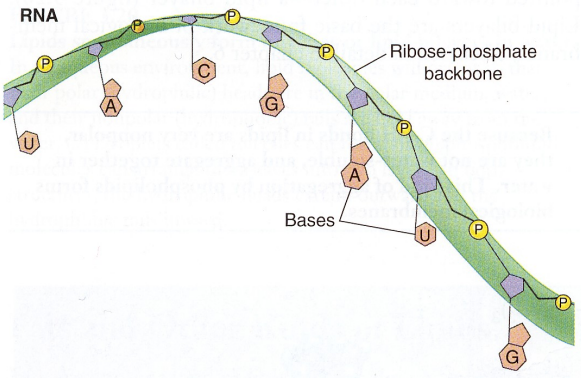


Figure 4.3: The structure of the RNA molecule [RAVE05]

Messenger RNA from the nucleus is created as a transcripted copy of portions of the DNA. This transcript is passed out from the nucleus into the rest of the cell, where it serves as a blueprint specifying a protein's amino acid sequence. Ribosomal RNA is a component of the ribosomes. The ribosomes are structural units capable of synthesizing a protein from the information carried by an mRNA. Transfer RNA is used by the ribosomes to help synthesize proteins.

RNA will be described in greater detail in chapter 4.2.

4.1.2 The Protein

Proteins are complex and versatile molecules carrying out a diverse array of functions, including defence, transport, support, motion, regulation, and storage. They are composed of one or more long chains (polypeptides) of up to 20 different amino acids linked by peptide bonds. An amino acid is a molecule which contains an amino group (NH2), a carboxyl group (COOH), and a hydrogen atom, all bonded to a central carbon atom. Each amino acid also has a side group which determines the amino acid's chemical properties. Although many different amino acids occur in nature, only 20 commonly occur in proteins.

The shape of the protein is very important because it determines the protein's function. Proteins consist of long amino acid chains folded into complex shapes. The structure of the protein has traditionally been discussed in terms of four levels of structure: primary, secondary, tertiary and quaternary. Lately an additional two levels of structure are increasingly distinguished by molecular biologists, namely motifs and domains. This next section briefly describes each of the structural levels. An overview is given in figure 4.4.

Primary structure. The specific amino acid sequence of a protein is its primary structure. This is determined by the nucleotide sequence of the gene that encodes the protein.

Secondary structure. Hydrogen bonding between the polar groups of the main chain decides the secondary structure. The hydrogen bonding results in two different structures: the alpha helix and the beta pleated sheet.

Motifs. The elements of the secondary structure can combine to proteins in characteristic ways. These combinations are called motifs. One very common motif is beta-alpha-beta, named “Rossman fold”

Tertiary structure. The final folded shape of a protein, which positions the various motifs and folds non-polar side groups into the interior, is called a protein's tertiary structure. The stability of a protein, once it has folded into its 3D shape, is strongly influenced by how well its interior fits together.

Domains. A structurally independent functional unit in the protein is called a domain. A single polypeptide chain connects the various domains, like a rope tied into several adjacent knots.

Quaternary structure. A protein's subunit arrangement is called its quaternary structure. When two or more polypeptide chains associate to form a functional protein, the individual chains are referred to as subunits of the protein.

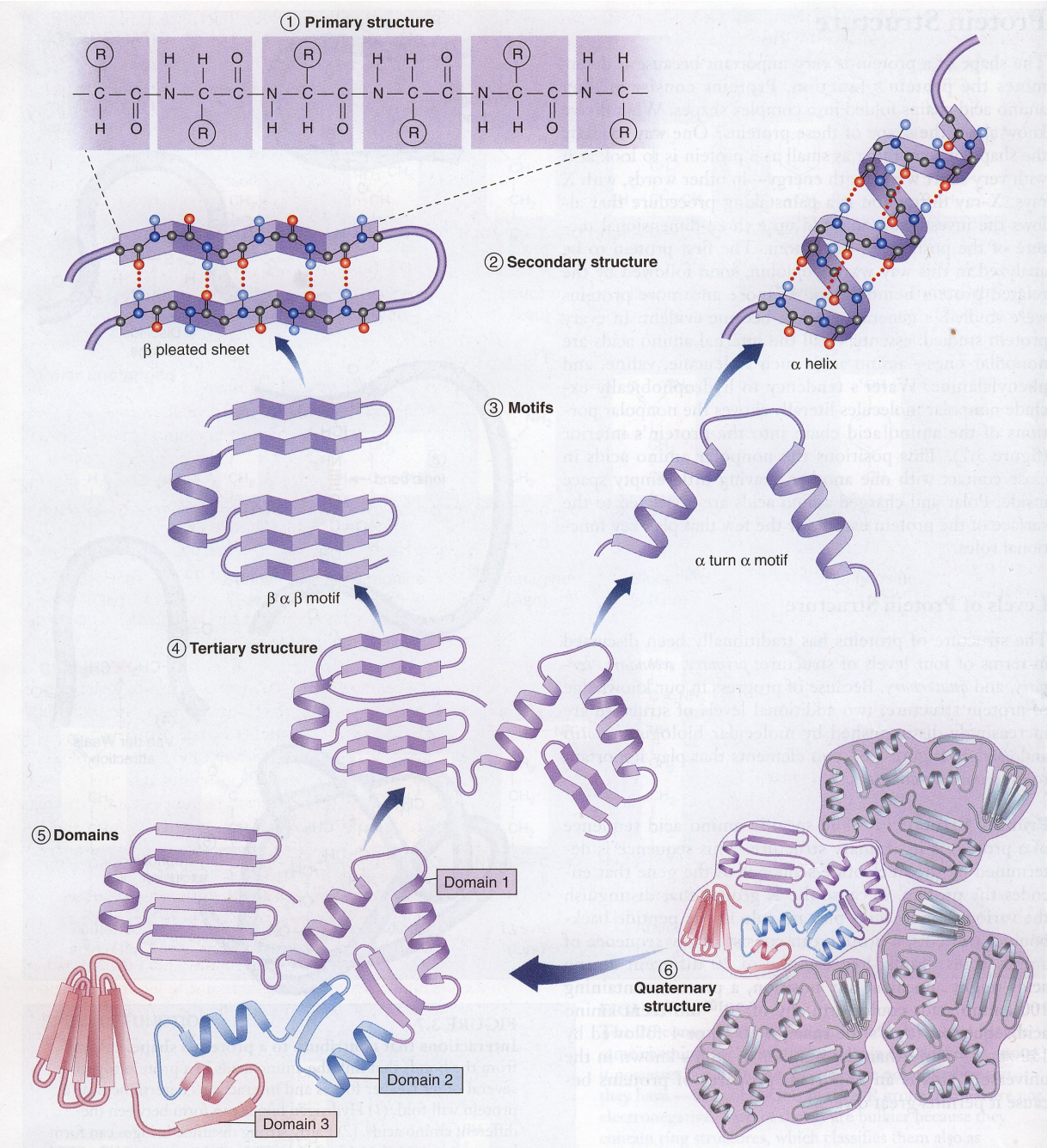


Figure 4.4: The six levels of structure in a protein [RAVE05]

4.1.3 The Cell

A cell is a membrane-bound unit containing, among other things, DNA and proteins. Cells are so small, typically 10-100 micrometers, that their existence were not discovered until 1665, when Robert Hooke used a self-built microscope to examine a thin slice of cork. He found in it a honeycomb organization of tiny, empty compartments. The compartments were empty

because the cells were dead. He termed the compartments *cellulae* (Latin, "small rooms"). This term has later come down to us as *cells*.

The Cell Theory

The modernized form of the cell theory includes the following principles:

1. All living organisms are composed of one or more cells, and the life processes of metabolism and heredity occur within these cells.
2. Cells are the smallest living things, the basic unit of organization in all organisms.
3. Cells arise only by division of a previously existing cell.

The third principle raises an interesting question: how did the very first cell come into existence? This touches on a highly debated issue: the origin of life. There is no definite answer to this question. Many possible solutions have been put forward, some with more merit than others. Perhaps the most widely accepted theory nowadays is "the RNA world", which basically states that given the right environment a cell may spontaneously arise [DUVE95]. This is an extremely oversimplified explanation of the theory, but a further discussion on this topic would clearly be outside the scope of this thesis.

Two Types of Cells

Cells can be separated into two major groups, *procaryotes* and *eucaryotes*.

Procaryotes are the simplest of the known organisms. They consist of cytoplasm surrounded by a plasma membrane and encased within a rigid cell wall with no distinct interior compartments – see figure 4.5. Most procaryotes have no membrane-bounded organelles (an organelle is any structure within the cell with a specialised function), nor do they have a true nucleus. The DNA, coiled up in a loop, floats freely inside the cell. Some of the procaryotes also have one (or more) flagella which they use to move.

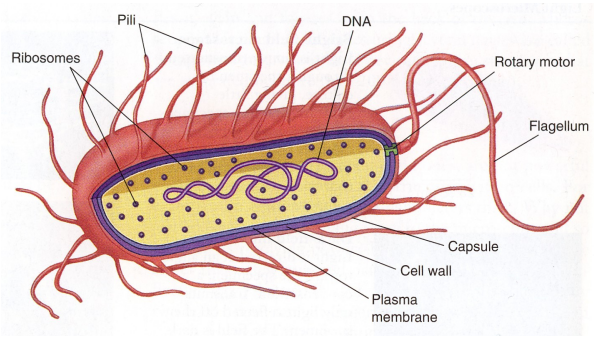


Figure 4.5: Procaryotes are simple organisms with no distinct interior compartments. [RAVE05]

Eucaryotes are the far more complex counterpart to the procaryotes – see figure 4.6. They are compartmentalized, which means multiple biochemical processes can proceed simultaneously

and independently. This compartmentalization is achieved by the endomembrane system and by numerous organelles. Eucaryotic cells also contain vesicles; smaller sacks used to store and transport a variety of materials. One of the major differences from procaryotic cells is the separation of the genome (the DNA) from the rest of the cell by a nuclear envelope. The DNA is wound tightly around proteins, packaged into compact units called chromosomes, and stored inside the nuclear envelope. While the procaryotes are characterized by a strong cell wall, many eucaryotes lack this feature. They maintain their structure trough an internal protein scaffold, the cytoskeleton, which all eucaryotes possess.

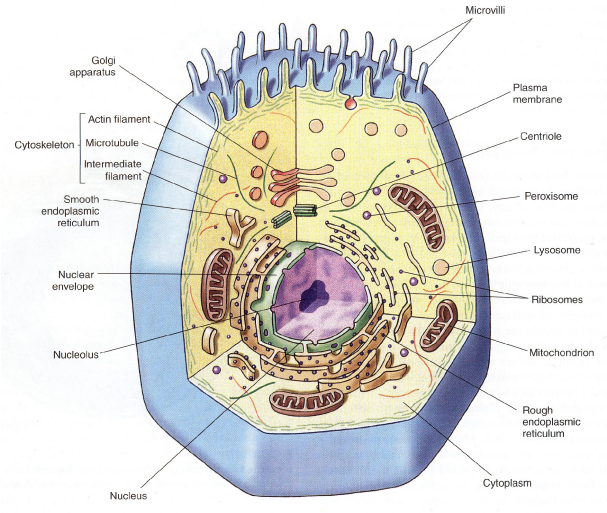


Figure 4.6: The eucaryotic cell is far more complicated than the procaryotic [RAVE05]

As the focus in this thesis is on modelling the development processes as it occurs in eucaryotes, procaryotes will not be further discussed. There are two main reasons for focusing solely on eucaryotes: first of all, procaryotes are unicellular organisms, making them unsuitable as models for growing a multicellular organism. Second, the gene regulatory network in eucaryotic cells are far more interesting (the existence of introns and exons is one example), enabling complex interactions

When writing about cells, eucaryotic cells are to be understood, unless clearly stated otherwise.

Structure

The cell consists of three major parts: the *plasma membrane*, the *cytoplasm* and the *nucleus*

Plasma membrane. The plasma membrane encloses a cell and separates its contents from its surroundings – see figure 4.7. It is a phospholipid bilayer, about 5-10 nanometers thick, and embedded with proteins.

These proteins in the plasma membrane are largely responsible for the cell's ability to interact with its environment. Transport proteins help molecules and ions move through the plasma

membrane, either from the surrounding environment to the interior of the cell or vice versa. Receptor proteins induce changes within the cell when they come in contact with specific molecules in the environment, such as hormones, or with molecules on the surface of neighbouring cells. These molecules can function as markers that identify the cell as a particular type. This interaction between cell surface molecules is especially important in multicellular organisms, whose cells must be able to recognize each other as they form tissue.

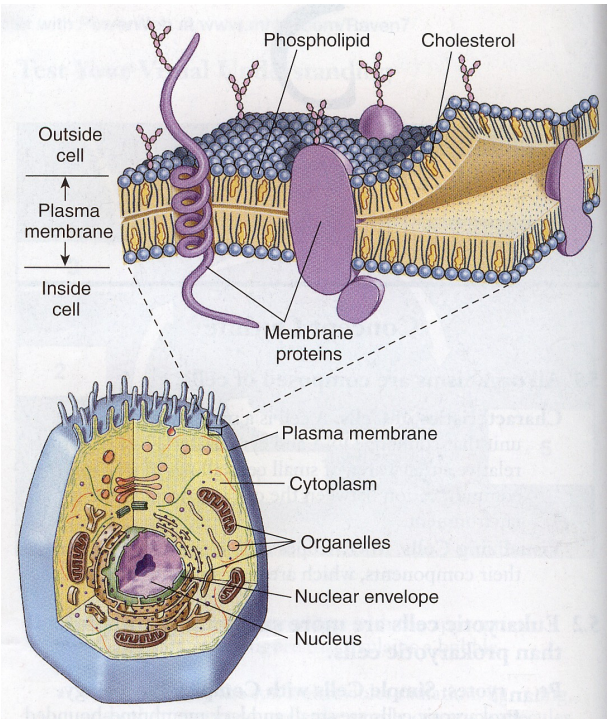


Figure 4.7: The plasma membrane separates the interior of the cell from its environment [RAVE05]

Cytoplasm. A semifluid matrix called the cytoplasm fills the interior of the cell, exclusive of the nucleus lying within it. The cytoplasm contains sugar, amino acids, proteins and organelles.

Nucleus. The nucleus is the "information centre" of the cell – see figure 4.8. This is the repository of the genetic information (DNA). The nucleus is roughly spherical in shape, and in animal cells, it is typically located in the central region of the cell. Most eucaryotic cells possess only one nucleus, but some have two or more.

The nucleus is wrapped in two phospholipid bilayer membranes, separating it from the rest of the cell. This double layered membrane, called *the nuclear envelope*, is crowded with shallow depressions called *nuclear pores*. These pores are filled with proteins that act as gatekeepers, permitting certain molecules to pass in and out of the nucleus.

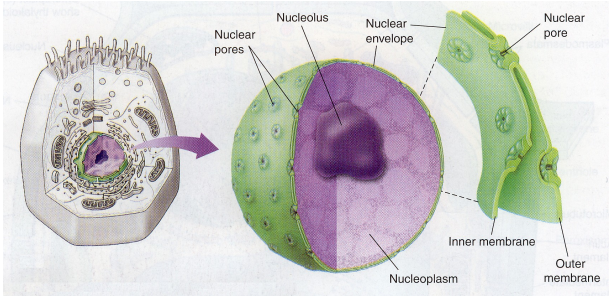


Figure 4.8: The nucleus is where the DNA is located in eucaryotic cells [RAVE05]

4.2 Protein Synthesis - From DNA to Protein

The biochemical activity of a cell depends on production of a large number of proteins, each with a specific sequence of amino acids. The ability to produce the correct proteins is passed between generations of organisms, even though the protein molecules themselves are not. DNA encodes information which specifies the sequences of amino acids that makes up the proteins in a cell. Because the two chains in the DNA double helix are complementary to each other, and because DNA replication is semiconservative, DNA is able to create exact copies of itself. The DNA replication is semiconservative because each of the two strands of the DNA will become a new DNA, using itself as a template to build the new DNA. This allows for all the cells in an organism to carry DNA encoding the same information, and hence for the organism to pass this information on to new generations via gametes (egg or sperm cell).

4.2.1 Gene Expression

All organisms, from the simplest bacteria to humans, use the same basic mechanism of reading and expressing genes. This is a mechanism so fundamental to life as we know it that it is often referred to as the Central Dogma: information passes from the genes (DNA) to an RNA copy of the gene, and the RNA copy directs the sequential assembly of a chain of amino acids – see figure 4.9. The two steps of the Central Dogma, taken together, are a concise summary of the events involved in the expression of an active gene. Biologists refer to this process as *gene expression*. The implementation of the development simulator will relay heavily on this mechanism.

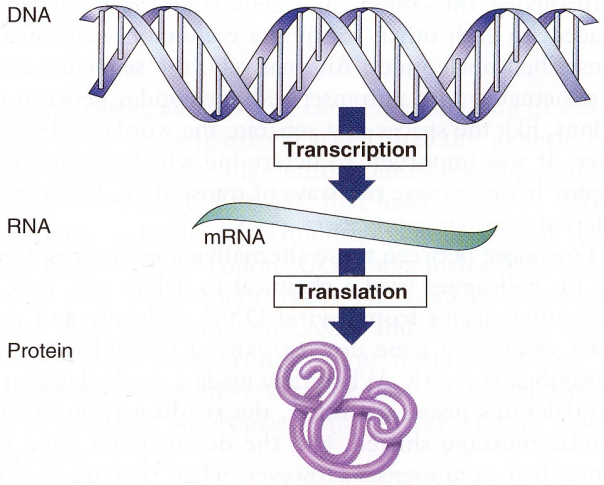


Figure 4.9: During gene expression DNA is transcribed to a strand of mRNA, which in turn is translated to a protein [RAVE05]

Within genes that encode proteins, the nucleotide sequence of DNA is read in blocks of three consecutive nucleotides. Each block, or codon, codes for one amino acid.

Transcription

The first step of the Central Dogma is the transfer of information from DNA to RNA. This occurs when an mRNA copy of the gene is produced. Because the DNA sequence in the gene is transcribed into an RNA sequence, this stage is called transcription.

Only one of the two strands of DNA is transcribed. This strand is called the template strand, while the other is called the coding strand. Transcription starts at RNA polymerase binding sites called *promoters* on the template strand. A promoter is a short sequence that is not itself transcribed by the polymerase that binds to it. RNA polymerase is an enzyme which carries out the transcription. In procaryotes there is only one RNA polymerase, while in eucaryotes there are three; RNA polymerase I, RNA polymerase II, and RNA polymerase III. They are specialized to transcribe rRNA (ribosomal RNA), mRNA (messenger RNA) and tRNA (transfer RNA), respectively. Of these three, only RNA pol II is relevant for this thesis, so when writing about transcription, the transcription of mRNA by RNA pol II is to be understood.

The transcription process is accomplished in three phases: *initiation*, *elongation* and *termination*.

Initiation. Transcription is initiated when a set of transcription factors (proteins) recognizes a promoter and binds to it. RNA pol II then associates with the transcription factors and the DNA, forming the initiation complex – see figure 4.10. A segment of the DNA helix is now unwound and opened up by RNA pol II, creating what is called a *transcription bubble*. The stage for the assembly of the mRNA chain is now set.

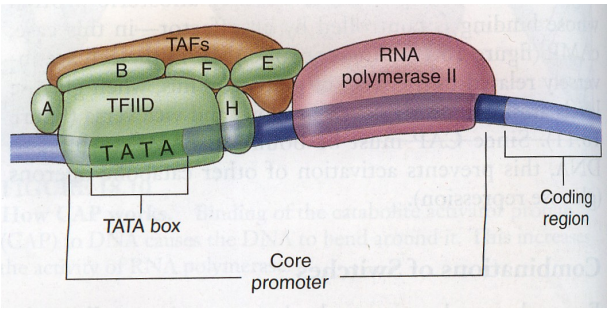


Figure 4.10: The formation of the initiation complex at a gene's promoter [RAVE05]

Elongation. Inside the transcription bubble the two strands of DNA are exposed. The synthesis of mRNA takes place at the exposed template strand where nucleotide building blocks are assembled into an mRNA chain. The transcription bubble now moves along the DNA strands into the gene. RNA pol II transcribes the DNA by adding the corresponding complementary nucleotide to the growing mRNA strand as it encounters each DNA nucleotide – see figure 4.11. The DNA is unwound as it enters the RNA pol II, transcribed into mRNA inside, and finally rewound when it leaves. This process continues until the whole gene is transcribed.

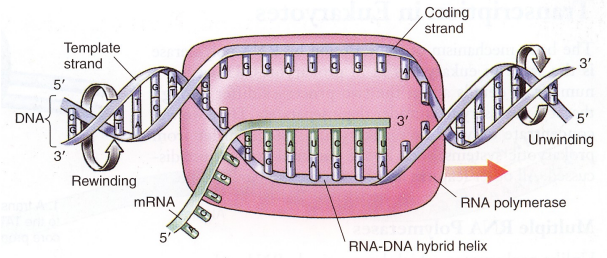


Figure 4.11: The synthesis of mRNA takes place inside the transcription bubble [RAVE05]

Termination. When the RNA pol II arrives at a transcriptional "stop" signal at the end of the gene, it disengages from the DNA and releases the newly assembled mRNA chain. An example of a simple "stop" signal is the *GC hairpin*, a series of G-C base-pairs followed by a series of A-T base-pairs

RNA polymerase has no proofreading capabilities, so transcription errors do occur. However, most genes are transcribed many times, so a few faulty mRNA will not be harmful.

Translation

The second step of the Central Dogma is the transfer of information from RNA to protein, which occurs when the information contained in the mRNA transcript is used to direct the sequence of amino acids assembled during the synthesis of polypeptides. This process is

called translation because the nucleotide sequence of the mRNA transcript is translated into an amino acid sequence in the polypeptide.

The translation of mRNA into protein is accomplished in large RNA-protein aggregates named *ribosomes,* located in the cytoplasm. Activation enzymes capable of linking tRNA molecules to specific amino acids, are doing the actual translation. The ribosome uses mRNA to place the tRNA molecules in correct sequence, and then the amino acids connected to the tRNAs are linked to each other, forming a polypeptide chain.

Ribosomes consist of two major parts: the small and the large ribosomal subunits. They have three sites used in the assembly of a polypeptide chain: the A site (for aminoacyl) where the tRNA will bind, the P site (for peptidyl) where peptide bonds will form, and the E site (for exit) where empty tRNA will exit the ribosome.

The translation is done in four phases: *initiation*, *elongation*, *translocation* and *termination*.

Initiation. A tRNA (with anticodon UAC) linked to the amino acid methionine binds to the small ribosomal subunit at the P site. It is helped into the correct position by proteins called initiation factors. This forms the initiation complex. The initiation complex, guided by another initiation factor, then binds to the “start” sequence (the codon AUG) on the mRNA. This initiation process is illustrated in figure 4.12.

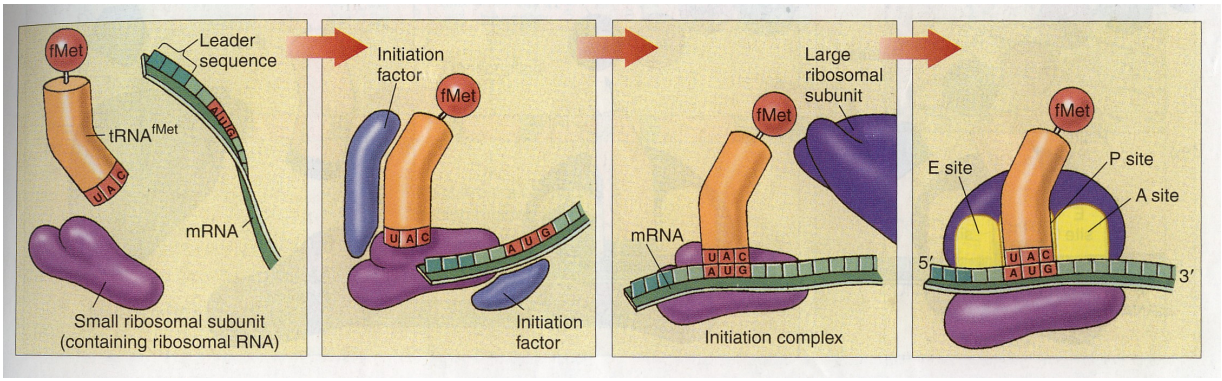


Figure 4.12: The initiation complex binds to the "start" sequence on the mRNA [RAVE05]

Elongation. Once the initialization complex has formed, the large ribosomal subunit binds to it and the codon next to AUG is exposed at the A site. This makes it possible for a tRNA with the appropriate anticodon to bind to the exposed codon. A peptide bond between the first and second amino acids is created, and the link between the tRNA at site P and its amino acid is broken. Elongation is illustrated in figure 4.13.

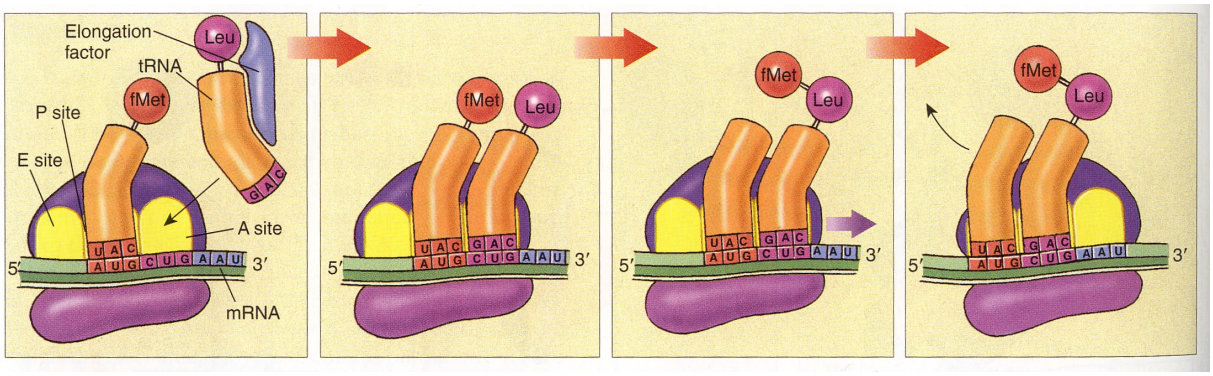


Figure 4.13: Elongation is the actual translation from mRNA to protein. This is the step where the correct amino acid is added to the growing peptide chain, as guided by the mRNA transcript [RAVE05]

Translocation. The ribosome now moves (translocates) three nucleotides along the mRNA, moving the tRNA at site P to site E, the tRNA at site A to site P, and exposes the next codon at site A. Again, a tRNA with the appropriate anticodon may bind to this codon. This process is repeated, translating one codon at a time.

Termination. Elongation and translocation continue in this fashion until the ribosome encounters a translational "stop" signal (nonsense codon). Nonsense codons do not bind to tRNA, but are recognized by proteins called *release factors*, which releases the polypeptide from the ribosome. A new protein has now been created.

4.2.2 Control of Gene Expression

The ability to control gene expression in a cell is an essential mechanism to all organisms. It is critical both in directing development and maintaining homeostasis (constant internal environment) in the cells.

Gene expression can be controlled at six levels – see figure 4.14:

1. Initiation of transcription
2. RNA splicing
3. Passage through the nuclear membrane
4. Destruction of the transcript
5. Protein synthesis
6. Post-translational modification

Initiation of transcription is referred to as the *transcriptional* level, while levels 2-6 are collectively referred to as the *posttranscriptional* levels.

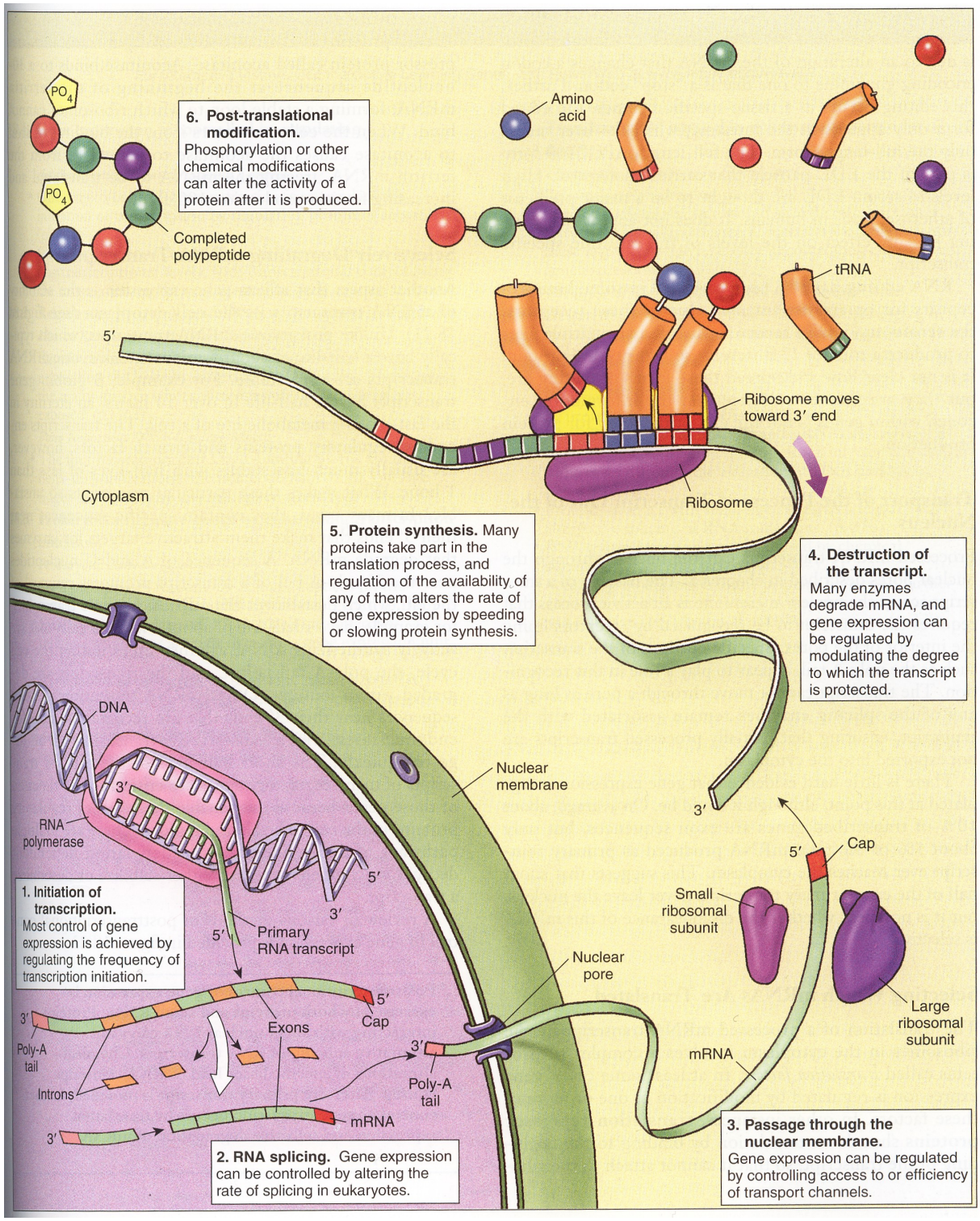


Figure 4.14: The six levels where gene expression can be controlled [RAVE05]

4.3 Transcriptional Control

This is the most common form of gene expression. Basically it is a control of which genes in the DNA are transcribed to mRNA and which are not.

For transcription to take place a variety of proteins (factors) is needed. These factors fall into two categories: the basal transcription factors and the specific transcription factors. The former is necessary for the assembly of the initiation complex at a promoter (see chapter 4.2.1), while the latter increases the level of transcription in certain cell types or in response to specific signals.

Basal factors, while necessary for transcription to occur, are not capable of raising the level of transcription above the basal level. When higher levels of transcription is required, specific factors, called *activators*, are needed.

Activators work by binding to the DNA at regions called *enhancers*. Activators and enhancers act in a position- and orientation-independent manner, which means the enhancer need not be next to the gene it affects. When bound to the enhancer, the activator interacts with transcript factors associated with RNA pol II (when present at the promoter), thereby increasing the level of transcription – see figure 4.15.

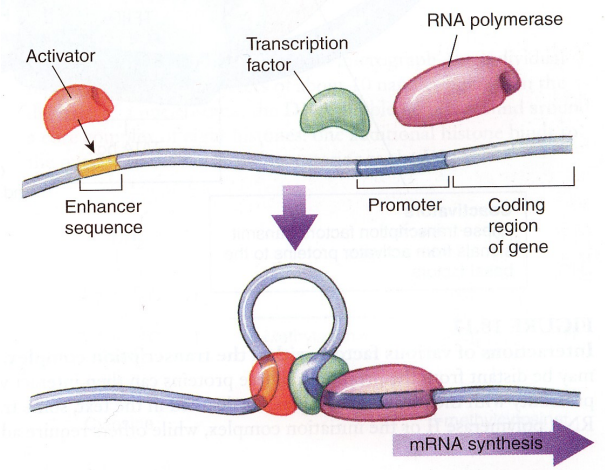


Figure 4.15: Activators work in a position- and orientation independent manner, and are capable of raising the level of transcription [RAVE05]

The DNA is divided into several linear units called chromosomes. Histones are packaging proteins associated with these chromosomes. The chromosomes are coiled up around clusters of histones called nucleosomes. These again are organized into higher-order structures. Nucleosomes are able to block binding of transcription factors and RNA pol II at the promoter, while higher-order structures are able to make promoters more or less accessible by modifying the histones. Gene regulation at higher-order structures is, however, not yet completely understood.

Taken together, all this contributes to a great flexibility in the control of gene expression.

Posttranscriptional Control

The unaltered mRNA transcript of a gene is called the primary transcript. In general, posttranscriptional control processes involve the recognition of specific sequences on the primary transcript. The primary transcript is composed of numerous short coding sequences (exons) embedded within long stretches of noncoding sequences (introns).

The first and major point of control is RNA splicing where the introns are removed by a spliceosome. The remaining exons can be spliced together in different ways, allowing different proteins to be assembled from the same gene – see figure 4.16. Editing the mRNA is also a possibility. This involves chemical modification (deamination) of a base to change its base-pairing properties, cytosine to uracil or adenine to inosine (pairs as G during translation).

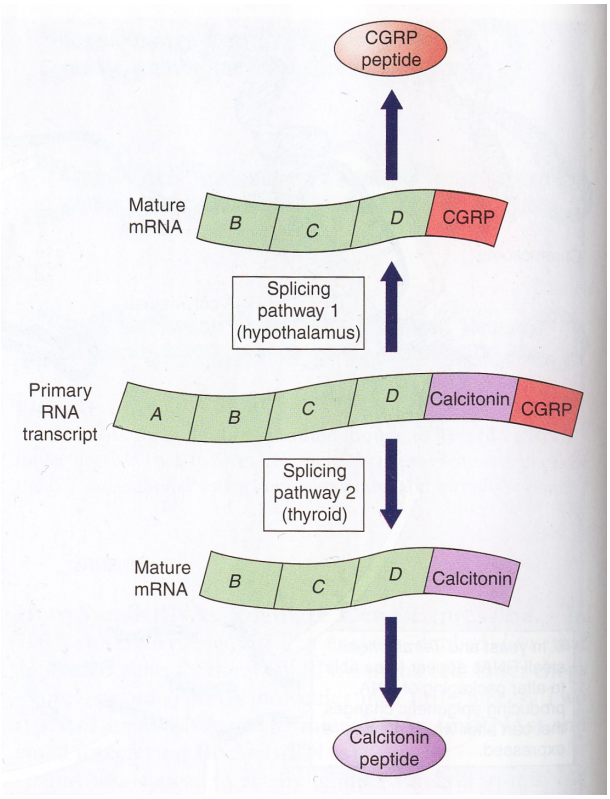


Figure 4.16: Example of two different ways of splicing a primary mRNA transcript, creating either a CGRP peptide or a Calcitonin peptide [RAVE05]

The mRNA is not allowed to leave the nucleus before the splicing is completed. The nuclear pores may regulate if, and how efficiently, the mRNA is exported into the cytoplasm. There is little hard evidence for this actually happening, but some support has been found.

The degree of protection on the mRNA decides how long it will survive in the cytoplasm. Increasing protection makes the mRNA survive longer, and hence more likely to be translated.

The translation of mRNA by the ribosomes requires a variety of proteins. Increasing or decreasing the availability of these proteins is another way to control gene expression.

When the mRNA has been translated, the resulting protein may be further altered in a process called post-translational modification. This involves chemical modification which may alter properties like activity or stability.

An overview of the basic elements in biological development has now been given. But so far, the discussion has been kept on the cellular level. How are these basic elements related to the growth and shaping of a multicellular organism? This is the topic of the next chapter, where the development mechanisms will be discussed.

4.4 Development Mechanisms

In biology, development refers to the process of growing and developing the fertilized egg into a full-grown organism. Three important mechanisms involved in this process are *cell growth*, *differentiation* and *morphogenesis*. These mechanisms do overlap and should not be viewed as independent mechanisms working alone. Rather, they depend in great deal on each other, working together to achieve successful development. They will now each be described in greater detail.

4.4.1 Cell Growth – Division of Cells

The term “cell growth” is used in two different ways in biology. One is the reference to an increase in the cell's size, while the other is a shorthand for the idea of growth in cell numbers by means of cell reproduction. The focus in this section will be on the latter.

The reproduction of a cell is a process in which the cell, called the mother cell, is divided into two daughter cells, passing along genetic material. This is called *cell division*. There are two kinds of cell divisions: *meiosis* and *mitosis*. Meiosis is a reduction division process in which the number of chromosomes in certain cells is halved during gamete (egg and sperm cells) formation. This kind of cell division is not relevant for this thesis, and will not be discussed further. Mitotic cell division involves nuclear division in which replicated chromosomes separate to form two genetically identical daughter nuclei. Followed by a division of the cell's cytoplasm, a process called cytokinesis, this results in two identical daughter cells.

4.5 The Cell Cycle

Mitosis is a key phase of what is called the *cell cycle*. The cell cycle is a process of growth and division which is repeated over and over in most eucaryotic cells. The length of this cycle varies considerably; from a few minutes to several years. The cell cycle can be divided into three major phases: interphase, mitosis and cytokinesis – see figure 4.17.

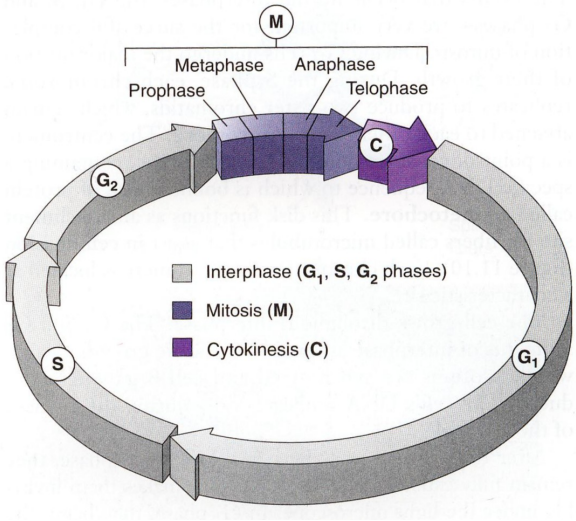


Figure 4.17: The cell cycle consists of three phases: interphase (G_1, S and G_2), mitosis (M) and cytokinesis (C) [RAVE05]

Interphase. This is the portion of the cell cycle between cell divisions. The interphase consists of three sub phases; G_1, S and G_2. G_1 is the primary growth phase of the cell. S is the phase in which the cell synthesizes a replica of the genome. G_2 is the second growth phase, in which preparations are made for genomic separation. During interphase the chromosomes are dispersed in the nucleus and appear as a network of threadlike strands.

Mitosis. During mitosis the nuclear division takes place. This is done in four phases, which occur without interruption: prophase, metaphase, anaphase and telophase. When the prophase begins, the individual condensed chromosomes become visible with an ordinary light microscope. The nuclear envelope breaks down, the spindle apparatus (an array of microtubular spindle fibers) is assembled, and the gene expression stops. More microtubules appears, growing from the two poles towards the chromosomes, linking sister chromatids to opposite poles. During metaphase the chromosomes align in the centre of the cell. This alignment is called the metaphase plate and is an indication of the future axis of cell division. During anaphase the sister chromatids are pulled apart by the microtubules, and during telophase the spindle apparatus is disassembled, nuclear envelopes are reestablished, and normal expression of genes is once again initiated.

Cytokinesis. The process of physically dividing the cytoplasm of a mother cell to create two daughter cells is call cytokinesis. The two nucleus created by mitosis is now arranged in the mother cell in such a way that each of them will will end up in one of the two daughter cells. The number of organelles going to each daughter cell is not controlled, but as long as some of each organelle are present in each daughter cell, the organelles can replicate to reach the appropriate number.

Control of the cell cycle

The cell cycle can be put on hold at different points, called checkpoints. At any of these checkpoints the process can be checked for errors and, if necessary, halted. This makes the cell responsive to both its internal state as well as signals from the environment. There are tree of these checkpoints: G_1/S, G_2/M, and at the spindle assembly. G_1/S is the primary point at which the cell "decides" to divide or not. Once the decision is made, the cell is committed to divide. The next checkpoint is G_2/M, where the success of DNA replication is assessed. The spindle assembly is in late metaphase. It ensures that all of the chromosomes are attached to the spindle in preparation for anaphase.

4.5.1 Differentiation

Differentiation is a developmental process by which an unspecialised cell undergoes a progressive change to a more specialized form or function. Put more simply: it is the process by which cells acquire a type. It is through this process the forming of different cell types in different parts and tissues of an organism is possible.

The DNA in a cell will, with a few exceptions (like the blood cells, which have no DNA), stay the same even after differentiation have taken place. This was shown by the cloning of the sheep Dolly, where a fully differentiated cell was able to develop into a full-grown sheep [CAMP96].

In vertebrates, all cells up until the eight-cell stage (after the three first cell divisions) in the developing organism are said to be *totipotent*. This means that they have the potential to express all genes in their DNA, and hence the ability to become any of the cell types found in the full-grown organism. At the eight-cell stage, the pathway that will influence the future development fate of the cells (i.e. which cell types they will eventually differentiate to) is determined. The commitment of a particular cell to a specialized development path is called *determination*.

How do the cells go from being totipotent to being committed to a special development path? Proteins called *gene regulatory proteins* are used to initiate developmental changes. When genes coding these proteins are activated, one of their effects is to reinforce their own activation. This makes the developmental switch deterministic, initiating a chain of events that leads down a particular development pathway. Often, before the cell becomes fully committed to a particular developmental pathway, it first becomes partially committed, acquiring positioning labels. These labels reflect the cell's location in the organism.

4.5.2 Morphogenesis

Morphogenesis is the shaping of tissues, organs and the entire organism. This encompasses the mechanisms which for example give humans the head at the right place, and makes us more than just a lump of cells. Many molecules are involved in morphogenesis, but three types are particularly important: *morphogens*, *transcription factor proteins*, and *cell adhesion molecules*.

Morphogens are soluble molecules that can diffuse and carry signals that control cell differentiation decisions in a concentration-dependent fashion. Transcription factor proteins are

a class of molecules that determine cell fate by interacting with the DNA, and by control of cell adhesion molecules a cell is able to migrate from one place to another.

Morphogenesis is achieved through different mechanisms, including *cell movement*, *induction*, *pattern formation* and *programmed cell death*.

Cell Movement

The migration of cells is important during many stages of development. Migration makes it possible for cells to rearrange themselves into specific structures. This arrangement is directed through both cell-to-cell interactions and cell-to-environment interactions.

Induction

In mammals, body form is determined by cell-to-cell interaction, a pattern called *regulative development*. Induction happens when a cell switches from one path to another as a result of interaction with an adjacent cell.

Pattern Formation

This is an unfolding process during development. In the later stages it may involve morphogenesis of organs, but during the earliest events of development, it lays down the basic body plan. The way this is achieved varies among different organisms, but some similarities do exist, like the establishment of the anterior-posterior (head to tail) axis and the dorsal-ventral (back to front) axis. This establishment of axis is possible because of an unequal distribution of developmental signals (called *determinants*) in the cell, creating a gradient. The gradient is created based on numerous factors, including control by maternally expressed genes (maternally produces mRNAs deposited in the egg cell), and entry point of the sperm cell.

Programmed cell death

Just like all living creatures, the life of a cell has a beginning and an end. However, both the beginning and ending of a cell's life are a bit different from that of most other living creatures. As stated earlier, cells arise only by division of a previously existing cell. This means a cell is not given birth to in the usual sense, but is “born” by division of the mother cell. When a cell dies, it either dies as a result of injury (necrosis) or because it is “programmed” to do so by its genome (apoptosis). When a cell dies because of necrosis it will typically swell and burst, releasing its contents into the extracellular fluid. Necrosis can be viewed as accidental death of cells, and therefore has no integral part in the development process. When dying of apoptosis cells shrivel and shrink, and their remains are taken up by surrounding cells. In contrast to necrosis, apoptosis plays an important role in the development of an organism, and all animal cells appear to possess this ability [RAVE05]. As an example of the impact apoptosis has on development, consider the fingers on the hand of a human; the cells between the fingers have died because of apoptosis, if not, humans would have paddles rather than digits.

5 Artificial Development and Related Work

Artificial development (AD) is in this thesis used as a term to describe a biologically inspired development process which performs the genotype-phenotype mapping in an evolutionary algorithm. The idea of an artificial development system can be traced back to the work of Mjolness et al in 1988 [MJOL88] and Kitano in 1990 [KITA90]. They proposed the first examples of evolving formal grammars in such a context. Their implementations did not involve cell division as they used matrix grammars instead. This approach has some drawbacks, as discussed in [GRUA92].

5.1 The Genotype To Phenotype Mapping Process

Kumar and Bentley uses the term embryogeny to describe the process guiding the genotype-phenotype mapping [BENT99]. They make a distinction between three different types of embryogenies: external, explicit and implicit.

5.1.1 External

External embryogenies are globally defined and they are external to the genotypes. These embryogenies are characterized by fixed, non-evolvable structures specifying how a phenotype should be constructed from the genotype. They are mainly designed by hand, and when designing, care must be taken to ensure it will always perform the desired function. An example of this type of embryogeny is Latham's system for evolving art [TODD92].

5.1.2 Explicit

Embryogenies where every step of the development process is explicitly stated are called explicit embryogenies. Although designing such an embryogeny by hand is possible, it is usually more practical to evolve it by means of genetic programming. This can be done by allowing the genotype and embryogeny to evolve simultaneously, making adaptive genotype-phenotype mapping possible. Broughton uses this type of embryology in his system for evolving architectural forms, where a Lindenmayer system is used as the embryogeny [BROU97].

5.1.3 Implicit

Using biological embryogeny as model, implicit embryogeny incorporate the important concepts conditional iteration, subroutines and parallel processing. This is in contrast to explicit embryogenies, where these concepts needs to be introduced manually. According to Holland, implicit embryogenies are types of constrained generating procedures which resemble neural nets, game theory and classifier systems [HOLL98]. By evolving a set of simple rules which can then be iteratively applied to each element of the growing solution, many large-scale problems can be tackled. An example of this is de Garis, who has had some success evolving CA-based implicit embryogenies to grow artificial neural nets on an immense scale [GARI94]

If using this categorization artificial development, as it is defined in this thesis, falls into the category implicit embryology.

5.2 Modelling Biology

What is there to gain from modelling development as it occurs in biology? According to Kumar and Bentley [KUMA03] the main benefit is that of construction: in biology, complex organisms are constructed, in the field of EA the knowledge of how to construct complex technology capable of adaptive, robust self-organization is sought.

They also suggest a number of advantages and disadvantages of using an algorithm based on development. The disadvantages include:

- Difficult to evolve by computer
- Difficult to analyse
- Difficult to create by hand
- Computationally expensive

Although these disadvantages are serious obstacles, there are many potential advantages, including:

- Reduction of the genotype
- Automatic emergence of complexity
- Compact genotypes defining complex phenotypes
- Repeated structure (subroutining, symmetry, segmentation)
- Adaptability
- Robustness to noise (fault tolerance)
- Regenerative capabilities
- Regulatory capabilities
- Able to help in understanding real biological processes and mechanisms

For an extensive explanation of these advantages and disadvantages regarding the use of a developmental algorithm, see [KUMA03].

5.3 Application areas and related work

The field of AD is relatively new. As earlier stated, the first known idea of an AD system was presented in [MJOL88] and [KITA90]. Various versions of such a system has been proposed and tested over the years, like Banzhafs et al's self-organizing binary string system [BANZ98], Dellaert and Beers system based on a random boolean network [DELL96], and the system of Cangelosi et al for developing neural networks, including both the mechanisms of cell division and of cell migration [CANG94]. The fact that this field is so new makes it difficult to get a structured overview of it. Additionally, the field has only been explored to a limited extent. This makes it often difficult to distinguish the promising approaches from the less promising ones.

The current application areas are limited, but the developmental approach has the potential for a much wider area of application. Current application areas include artificial neural networks, structure optimization and evolvable hardware. Researchers who have contributed, and some are still contributing, in these areas are Kitano [KITA90], Gruau [GRUA94], Miller [MILL03], Haddow and Tufte [HADD01][TUFT03], Gordon and Bentley [GORD02], and Sipper et al [SIPP97], to mention a few.

As already stated – see chapter 9, the research on AD can to some extent be segregated into two groups: those who make an effort to create a biologically plausible model and those who do not care that much about biological plausibility because their main goal does not require it. A presentation of some researchers in both these groups will now be presented.

5.3.1 Research Aimed at a Biologically Plausible Model of Development

Dellaert presented in his master's thesis from 1995 what he calls a biologically defensible development model [DELL95]. The model includes such biological concepts as the gene regulatory system, multicellular development, cell differentiation and neural development. The gene regulatory network is represented as a boolean network with genes and gene products as nodes and the connections between nodes. Gene products is a term used to encompass enzymes, proteins, receptors and other molecules participating in the gene regulatory network. The difference in activity among the various networks provides for differentiation of the cells. Dellaerts model is capable of generating a range of morphologies in a two dimensional grid world. It is also capable of evolving the gene regulatory system to optimize some performance function for the fully developed organism. Neural development was demonstrated by hand coding a functional agent, and the developmental process proved itself robust to changes in the morphology of an agent.

Eggenberger-Hotz is also one of those trying to model biological development in greater detail. He has designed systems capable of developing three dimensional organisms using such biological development mechanisms as gene regulation, cell division, cell death, cell differentiation and positional information [EGGE97]. Extensions of this basic system also incorporate cell induction [EGGE99], cell cleavage [EGGE03], and asymmetric cell division [EGGE04]. With these development systems he has been able to successfully demonstrate that it is possible, using these mechanisms, to develop shapes, patterns, organisms with regeneration ability (the ability to replace lost parts of itself), and moving creatures.

Kumar and Bentley are perhaps the two currently most profiled researchers trying to model biological development closely. Over the years they have experimented with different approaches. Their early research includes comparison between different types of developmental genotype-to-phenotype mapping in evolutionary algorithms. Three types of embryogenies were distinguished: external, explicit and implicit – see chapter 5.1. They found through experiments that implicit embryogeny outperformed, with respect to scalability, both the two other types of embryogeny and the standard one-to-one mapping. The experiments were performed using both tessellating tiles [BENT99] and letters of the alphabet [KUMA99] as targets. Later work include systems evolving three dimensional embryos. The first of these system that they implements had serious problems regarding evolvability, and experiments hinted that the representation used could be improved by allowing genotype redundancy (many-to-one mappings) along with placing similar solutions together in the solution space (allowing a more gradual evolution) [KUMA00]. A later system, the Evolutionary Developmental System (EDS), was aimed at modelling biology very close to learn more about which developmental mechanisms might be useful in an AD system. EDS included biological concepts such as embryos, cell, cell cytoplasm, cell wall, proteins, receptors, transcription factors, genes and cis-regulatory regions. The system develops the embryo in a isospatial coordinate system, instead of the more standard Cartesian, making the system biased towards developing more "natural" shapes. They experimented with the systems gene regulatory system, the shaping of embryo (morphogenesis), oriented cell division and cell differentiation [KUMA03b][KUMA03c][KUMA04b]. They found that it is possible to achieve morphogenesis without cell signalling, that the orientation of cell divisions may have major implications on the development, and that the EDS is capable of breaking symmetry.

5.3.2 Research Inspired by Biological Development

Miller does not consider it a goal to model biology closely in his development systems, even though he borrows numerous concepts from nature. He represents in [MILL03] and [MILL04] a system capable of growing a French flag organism in a two dimensional grid. The organism consists of cells with different colours (types). The cell's genotype is a representation of a feed-forward boolean circuit (the cell program). This cell program receives data regarding external environmental signals, the type of its own cell along with the cell type and chemical level for all neighbouring cells. The output of the cell program is a new chemical level, a new cell type (including dead and the current cell type), and growth (more than one direction at once is possible). The chemicals are allowed to diffuse. They do this according to a conservative diffusion rule which also ensures that the chemicals over time diffuse away from their origin. This development system was able to perform morphogenesis, self-repair and it showed adaptability to environmental changes. Morphogenesis was achieved in both producing a recognizable French flag which remained recognizable even after further development steps were introduced, and the successful development of a growing pattern of blue dots. Self repair was shown through partly recovery from severe damage of the French flag. The successful development of alternating red / blue dots showed the system was capable of adaption. Miller also found during his experiments with this system that although it was possible to develop organisms without chemicals, they were of much lower fitness and lacked stability. Miller also made contributions to the work of Vassilev [VASS00], and Liu and Tyrell [LIU05], both working towards evolution of digital circuits.

Bongard and Pfeifer created in [BONG01] and [BONG03] an artificial evolutionary system that uses a developmental encoding scheme to translate a given phenotype into a complete agent. This agent then acts in a physically realistic virtual environment to determine its fitness based on its performance on an assigned task. The genotype is in this system treated as a genetic regulatory network, and the changing expression patterns of this network over time leads to the growth of both the morphology and neural control of the multi-unit, articulate agent. The morphology of an agent is grown as follows: the genome (the genetic regulatory network) is placed inside a unit (roughly similar to the concept of a cell in other artificial development systems), the unit is injected with a small amount of gene products, and the genome is executed. Gene products are produced by genes and they do either have a direct effect on the phenotype or they regulate the expression of other genes. The initial injection of these gene products, where the gene products will be asymmetrically distributed, establishes a gradient which enables a breaking of shape symmetry in the developing agent. This is analogous to the establishment of the symmetry breaking morphogen gradient in a biological embryo. The unit is then allowed to grow, split and form connections to new units in a process regulated by the genome. The neural control is co-evolved with the morphology using cellular encoding [GRUA96]. Cellular encoding is a developmental method for evolving both the architecture and synaptic weights of a neural network. The evolved agents were assigned a task of either directed locomotion or block pushing. The results suggest that the designed system was sufficient to produce hierarchical, repeated structures, and that the inclusion of differential gene expression dissociated the information content of the genotype from the complexity of the phenotype.

Federici presents in [FEDE04] an extension of the model proposed by Miller [MILL03]. The model of Federici grows cells in a two dimensional grid. The cell growth is regulated by what he calls a Morpher: a standard feed-forward artificial neural network representing the gene regulatory system in a cell. This Morpher is what is evolved by the genetic algorithm part of this system. The system implements such biological concepts as cell type, internal metabolism and external chemicals. The cell type and the internal metabolism are both part of the cell, while the external chemicals is independent of the cells: it belongs to the environment. The Morpher receives data regarding the type of its own cell, the type of each of the four neighbour cells, the chemical diffusion, and the internal metabolism as inputs. The output includes cell production, chemical production, changing the cells type, and changes in internal metabolism. The development starts with one single cell containing a Morpher, which is allowed to execute its program, igniting the development process. Development is then allowed to continue for a specific number of steps, creating a pattern of cells. This pattern is then mapped to the hidden layer in an artificial neural network which is used in the control of a virtual agent. The fitness of an agent is calculated based on the its ability to collect food and avoid both poison and other agents. To enable a more incremental refinement of the Morpher, a novel method was introduced: embryonic stages. The idea behind embryonic stages is that each stage has control over a specific part of the development process. New stages are added incrementally, where those controlling earlier stages of development are developed first. When a new stage is introduced, it is created as a copy of the former stage, and the former stage is locked – it can not be changed any more. Later work with this system includes investigation regarding the effect of embryonic stages [FEDE04b] and investigation of system scalability and the robustness of the evolved phenotype [ROGG04][FEDE05b]. In these experiments, the system was assigned the task of evolving two dimensional patterns, like a Norwegian flag, instead of evolving a neuro-controller. This was done to make the system easier to analyse. The results were promising: the use of embryonic stages seems to increase evolvability, the system had

better scaling behaviour than its opponents (including direct encoding and a simpler development system aimed at hardware implementation), and the phenotypes created showed robustness towards changes, also when this feature was not selected for.

Dellaert and **Beer** extended the work of [DELL95] with the design of a much simpler and less biologically plausible system in [DELL96]. This system was able to develop, from scratch, complete autonomous agents that could perform simple tasks, like following a line. Dellaert has, since this work, moved from the field of development to do research in the areas of robotics and computer vision.

6 Goal

In this thesis an effort is made to design and test an artificial development (AD) system implementing a biologically inspired development process for the mapping of the genotype to the phenotype in an evolutionary algorithm (EA).

The objective of modelling biological development is two-folded: one objective regards computer science, while the other regards biology. The main reason for doing research on development from a computer science point of view is its potential to reduce the scalability problem troubling evolutionary algorithms. This is thought to be achievable through the introduction of a process of development to perform the genotype to phenotype mapping. The objective from a biological point of view is to gain a deeper understanding of the mechanisms at work during development, as this is a topic that is currently not completely understood by biologists. The focus in this thesis, however, will be on the former of these two; reducing the scalability problem.

As limited research has been performed in the field of artificial development a decision was made to create a novel system, providing an alternative to the current approaches, instead of performing a deeper investigation of one of the current systems.

Two systems were built during the work on this thesis: BioDev and ArtDev3D. BioDev was the first system designed and implemented, and was designed to model nature closely. The author had a participatory role in this project which was initiated and directed by Chen [CHEN04]. BioDev served as a source for insight into both the mechanisms of development and to which elements were crucial and which ones may be left out. This knowledge was later put to use in the design of the ArtDev3D system, the system which this thesis focus on. ArtDev3D was designed and implemented by the author alone, based on the experiences with BioDev.

The main goal in this thesis is to design an AD system implementing an effective development algorithm capable of reducing the scalability problem in EAs.

The scaling property will not be investigated directly in this thesis because of the time constraint. However, if the system is to be scalable it must also prove itself as useful on a smaller scale first. The way this is tested in this thesis, is by investigating two properties of the system. First of all, it is important that the system is capable of expressing a varied set of phenotypes. Secondly, it is important to be able to control the system, achieving the wanted effect when tweaking the system's parameters. Investigating these two properties lays the base for later testing of the system's scalability.

Effectiveness, in terms of the required computer power, is important if the algorithm is to be useful for practical application. ArtDev3D strives to achieve effectiveness, compared to BioDev, by reducing the complexity of the development process. A direct testing of effectiveness was not investigated in this thesis because of the time constraints.

One main goal, along with two sub goals have now been distinguished:

- Main goal – design an AD system implementing an effective development algorithm capable of reducing the scalability problem in EAs.

- Sub goal one – Ensure the development algorithm is flexible enough to develop a variety of different shapes

- Sub goal two – Ensure it is possible to control the system in such a way that tweaking it's parameter results in the wanted effects

Sub goals one and two are investigated in the experiments performed in chapters 10 and 11. These two sub goals lays the base for a throughout investigation of the main goal: the system's scalability.

Part II – The Development Model

7 BioDev – The Initial Model

My work on this master thesis started out in January 2004 as a teamwork between another master student, Yan Hua Chen, and myself. Together we designed and implemented a system supposed to "simulate artificial development using a biologically plausible model" [CHEN04] To reflect the fact that this system was supposed to model biological development closely, the system was given the name BioDev (Biological Development).

Although BioDev is not the system used to conduct the experiments described in this thesis, the system actually used, ArtDev3D, is heavily inspired by the BioDev system. Also, as designing and building BioDev was the initial part of my work on this master thesis, a brief description and discussion of the system will now be given.

7.1 The Model

BioDev consists of two major modules; the development module and the genetic programming module. The former takes care of the mapping from genotype to phenotype, while the latter is used to evolve a good approximation for the given target. The genetic programming module is a fairly standard implementation, and hence will not be discussed any further.

The major components of the model will now be described, followed by a description of its features.

7.1.1 The Organism

The organism is the developing lump of cell, from the zygote to the full-grown creature. It "lives" in an environment, called the universe, with definite space and time bounds.

The space the universe provides is a two dimensional grid. Each cell occupies exactly one location in the grid, and only one cell is allowed at each location.

The time is divided into discrete time steps, called clock ticks. The processes occurring in the universe are serialized within each clock tick, but in an effort to try to model the massive parallelism occurring in nature, processes executed within each clock tick, will not affect the organism until the end of that tick.

7.1.2 The Cell

The cells have a fixed size, and each cell occupies exactly one square in the universe grid. Inside each cell is a DNA, which is what decides how the cell will develop.

The cell can perform a variety of functions

- Cell division – the cell divides, creating two daughter cells.
- Cell death – the cell kills itself

- Movement – the cell can move a certain distance in the universe
- Signal emittance and reception – the cell is capable of sending an receiving signals from other cells

Each cell also has a precondition which must be fulfilled before the cell will perform any of these functions.

7.1.3 The DNA

The DNA consists of a number of genes. Each gene is composed of two parts: the regulatory region and the coding region.

The regulatory region contains both the information about whether the gene is blocked from transcription, and the promoter. The promoter is used when deciding if the gene should be transcribed.

The coding region contains all the information necessary to construct the protein the gene codes for.

7.1.4 The Protein

Each protein has a specific time to live, a function it performs, and a set of preconditions.

The time to live refers to how many clock ticks the protein is alive (active). When the protein dies, it is simply removed from the cell.

The function of a protein is the action it performs when activated. A protein may perform any of the following actions:

- Split – instructs the cell to divide
- Activate – transcribes genes from the DNA
- Apoptosis – instructs the cell to kill itself
- Block gene – blocks specific genes
- Unblock gene – unblocks specific genes
- Update the precondition of the cell – updates the precondition of the cell
- Send signal – sends a signal to specific target cells

The set of preconditions looks at the concentration of a certain type of protein, in addition to global and local location information. If all the preconditions are fulfilled, the protein is allowed to perform its action.

7.2 The Process of Development

The BioDev model focuses on four essential mechanism in development: growth, cell division, differentiation and morphogenesis. In nature, these mechanisms involves both the gene regulatory system, and cell communication. These two features are also implemented in BioDev, and will now be described. Following this comes a description of each of the development mechanisms.

7.2.1 The Gene Regulatory System

The “engine” behind the whole process of development is the gene regulatory system. Two important parts of the gene regulatory system are the expression of genes and the regulation of this.

Gene expression in BioDev is accomplished in one single step. This means that a gene is not first transcribed to mRNA which then is translated into a protein, as it is done in nature. Instead, the gene is directly transcribed to a protein. Also, one transcription protein (protein capable of transcribing genes) may, in this model, transcribe many genes simultaneously. This is not the way it works in nature, where a protein binds to a promoter when transcribing, making the protein unable to transcribe any other genes until it is finished transcribing the current one. The reason for the former implementation choice is to avoid what is thought to be unnecessary complexity. The latter choice was made because it allows for a smaller number of proteins than the biological approach to control the gene regulatory system, making the development process more efficient.

Regulation of gene expression occurs in nature at numerous places. In BioDev, regulation is restricted to happen in three ways. The first regulation mechanism is the production level of activator proteins in the cell. This depends on whether the specified preconditions are fulfilled. The second mechanism is the blocking and unblocking of genes, and the third is provided through the cell's precondition. The third mechanism is a particularly powerful one, as it makes it possible to activate or inhibit all gene expression in the cell at once.

7.2.2 Cell Communication

The signal proteins are responsible for all communication between cells. These proteins act as both hormones and chemical signal emitters. This enables these proteins to send signals both to adjacent cells and distant ones.

The receptor of a cell targeted by such signals is responsible for queuing and forwarding the signals to the interior of the cell, where they are handled along with the other actions supposed to take place in the cell.

7.2.3 Growth

As the cells have a fixed size, the growth of the organism is solely achieved through cell division. Inspired by the mechanism of contact inhibition in nature, a cell is only allowed to divide in a given direction if no cell is currently present there.

7.2.4 Differentiation

Differentiation is controlled through gene regulation. Two cells with different patterns of gene transcription and protein activation are, in this model, said to be differentiated from each other. This may happen through a change in the cell's precondition or the blocking of a set of genes, possibly induced by a received signal protein.

7.2.5 Morphogenesis

The positional information available to a cell in an evolving organism in nature, is in this model emulated through the use of what is called location information. Location information is used to encompass information on both a global (organism) and local (cell) level. The location information available to a cell includes the distance to both the centre and surface of the organism, the neighbours in a given direction, the neighbours within a given radius, the number of cells in the organism, and the height and width of the organism.

7.3 Results

A number of experiments where carried out, using different shapes as target. In addition, a set of fitness functions were tested. Some of the experiments were run with DNA coded by hand, while others were run using the GP to evolve the most fitting DNA.

Because of the time constraint, it was only possible to test BioDev for its capability to perform development. It was shown that the system was indeed able to develop differently shaped targets, starting from only one cell. In addition to this, some support for the properties of artificial development, as set forth by Kumar in his PhD thesis [KUMA04], was found. Examples of these are the possibility of a decentralized control, and the compactness of DNA in contrast to the full-grown organism. The interested reader is encouraged to read Kumar's thesis.

7.4 Limitations

What seems to be the greatest limitation of the BioDev system, is its inability to evolve good solutions. It has been shown, through hand coding of the DNA, that the development part of the system is capable of developing various shapes. However, when trying to evolve a given target using GP to find a usable DNA, the system experiences difficulties. A number of reasons for what causes this problem are suggested in [CHEN04].

Another type of problem troubling this implementation of BioDev is that the development process is very complex, with lots of interdependency between the various elements. This may in turn be a reason for the problem of achieving high fitness, because guiding evolution in correct direction is difficult when the discrepancy between the genotype space and the phenotype space is large. This is unfortunately not the only consequence of a complex development process. The time required to complete the process of developing a DNA to a full-grown organism is relatively large, making the system too time consuming to be of any practical use.

Because of these limitations, a decision was made to design a new system where the development process would be less complex. This new system was baptised ArtDev3D, and will now be described.

8 ArtDev3D – The new and improved model

As discussed in chapter 7.4, the BioDev system had become too complex, and it was also shown to be very difficult to evolve a good DNA to use for the development. A decision was made to design and implement a new system based on the experiences from BioDev.

One of the main difficulties when designing an artificial development system is to know which mechanisms in nature could be useful to include in it. Including every detail of biological development is out of the question for two reasons: all the details are simply not known, and even if they were, biological development is such an incredibly complex process that including every detail of it would just not be possible in practise. Working with BioDev gave a deeper insight into which mechanism could be useful and which ones not, than just theoretical knowledge could have given. This knowledge has been used to build what is believed to be a more effective and versatile artificial development system; the ArtDev3D.

ArtDev3D is an acronym for Artificial Development in 3D. The name was chosen to emphasize the change in focus from BioDev, where biological development was modelled closely, to a focus on an effective and simplified development process. The "3D" postfix denotes the change from a two dimensional organism to a three dimensional one.

As the ArtDev3D system is a novel model, it had to be designed and implemented from scratch. A lot of time and effort has been put into making the system both user-friendly (easy to understand and use), and extendible (including new mechanisms can easily be done by extending the framework).

8.1 System Overview

On a high abstraction level the system can be seen as having 3 layers: the artificial development layer (AD), the converter layer and the evolutionary algorithm layer (EA) – see figure 8.1. The converter layer is responsible for the proper conversions between the AD and the EA layers. The genotype from the EA is converted to a DNA, which is used to direct the development of the organism in the AD. The organism / phenotype is then evaluated by the fitness function, and the EA is updated with the evaluated fitness.

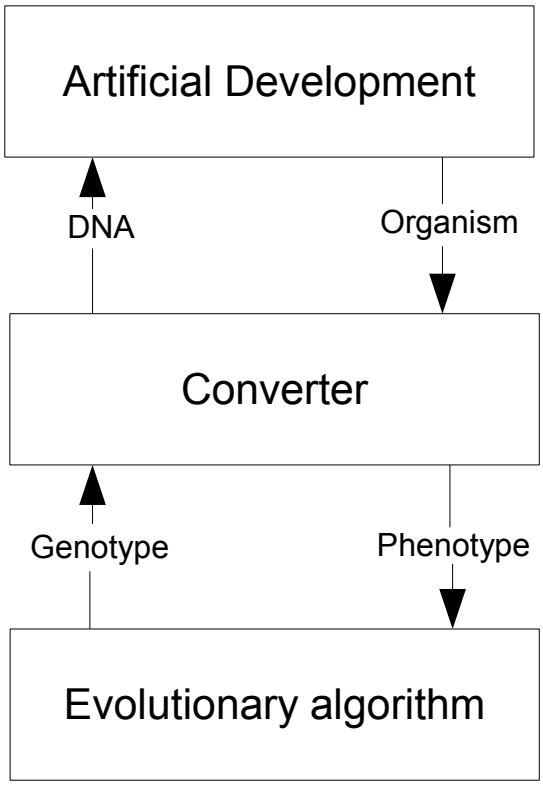


Figure 8.1: Design overview of the ArtDev3D system

This separation of the system into three distinct layers makes the AD completely independent from the EA. This is useful because it provides an easy way of testing different implementations of the EA in combination with the AD. The conversion layer will not be discussed as a stand-alone layer, as it, in practice, is an integrated part of the evolutionary algorithm.

8.2 The Evolutionary Algorithm

Genetic programming (GP) was chosen as the evolutionary algorithm to use in ArtDev3D. To make the algorithm better suited for the task, some variations to the classical GP – see chapter 3, were made. These variations concern both the way the genotype is represented and the way crossover and mutation are performed. The genotype representation will now be first described, followed by a discussion of the crossover and mutation operators, and finally the choice of fitness function and selection mechanism.

8.2.1 Genotype Representation

The genotype is represented in what can be viewed as a combination of variable length linear and tree structure – see figure 8.2. On the top layer, the genotype is a list structure of nodes.

The number of nodes in this list may increase or decrease during evolution. Each of these nodes is the root node in a tree structure representing a gene. The structure of these trees are rigid; all genes are represented as the same structure, and this structure cannot be changed.

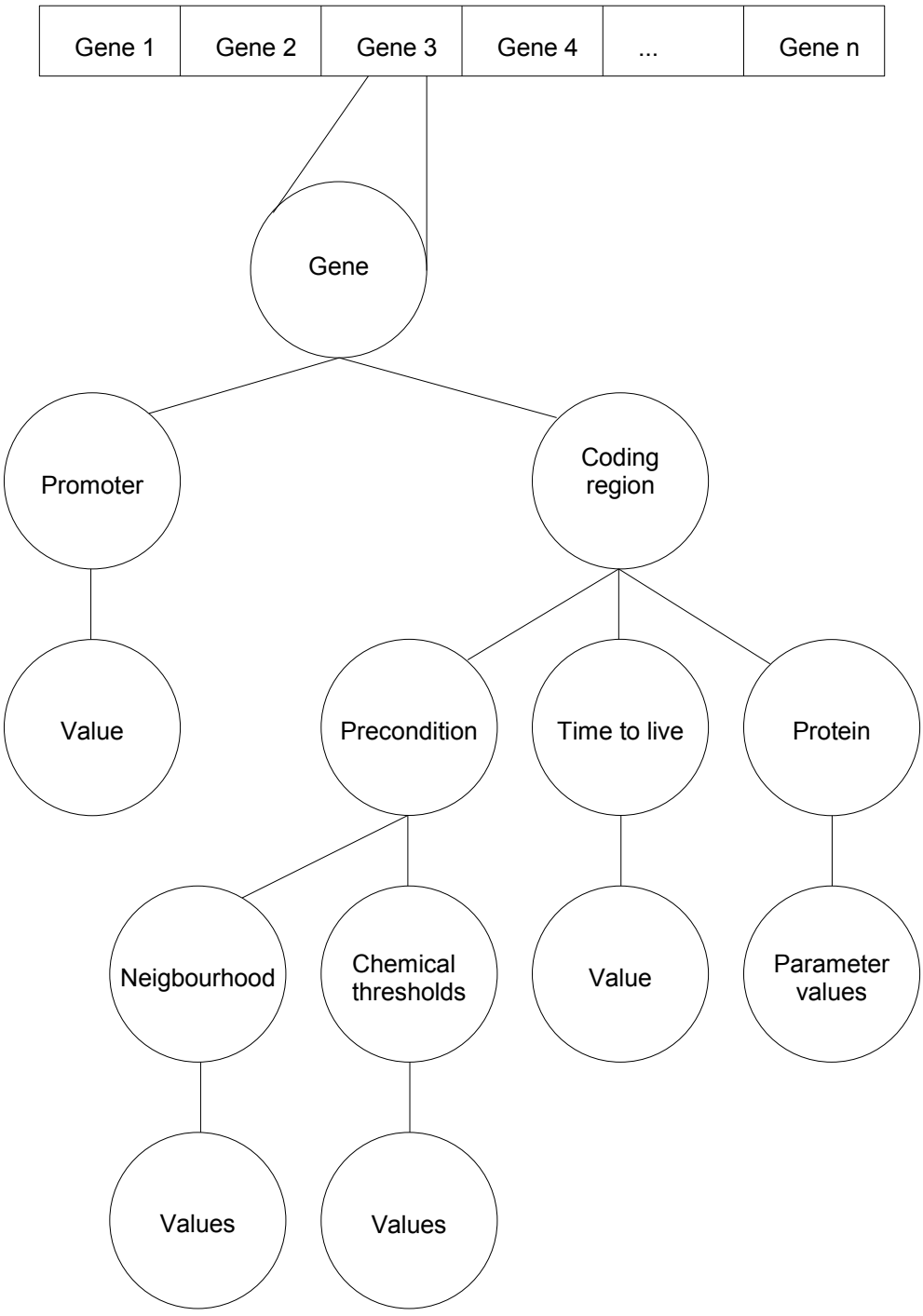


Figure 8.2: The structure of the genotype

Using the function and terminal terminology from chapter 3.1, the function set is the following: gene, promoter, coding region, precondition, neighbourhood, chemical thresholds, time to live and each of the different types of protein (change chemical concentration in cell, change cell type, transcribe genes, and divide cell). Each function in the function set has its own terminal set. These terminal sets are specified at run time.

8.2.2 Crossover

The crossover operator can be a very disruptive operator, but it is also what contributes to the spreading of good building block. To minimize the disruptive force, but at the same time allow good building blocks to spread, the crossover operator is only allowed to work on the top level of the genotype. This means that only whole genes, not sub trees within the genes, are exchanged between parents during crossover – see figure 8.3.

The crossover method used is two-point crossover. This method chooses two locations within the genotype at random and exchanges the information located between this two points with the other genotype. The variant used in ArtDev3D uses the same crossover points in both genotypes, so the number of genes exchanged are the same for both.

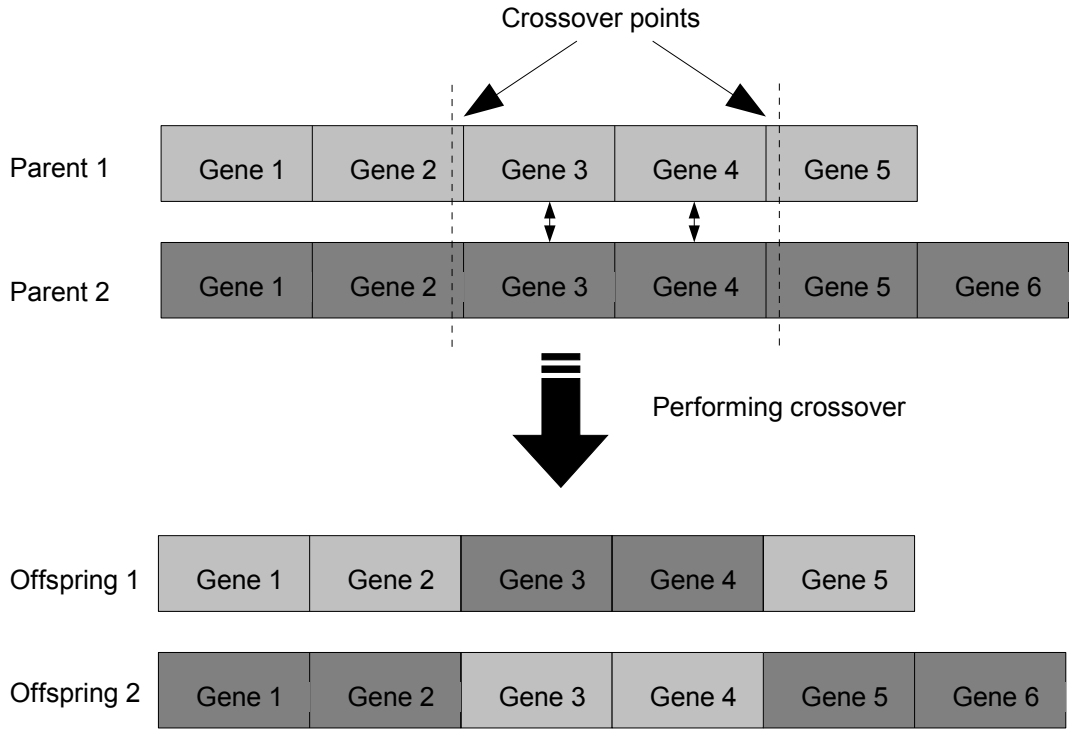


Figure 8.3: The crossover operator exchanges an array of genes between the two individuals it is applied to

8.2.3 Mutation

Two types of mutations are used in ArtDev3D: mutation at a node inside a gene, and duplication / removal of a gene. The former applies to the tree structure, while the latter applies to the linear top level of the genotype.

The mutation operator applied at the node level is a variation of the classic mutation operator – see chapter 3.2.3. However, this mutation operator has some limitations: only the terminals may be mutated, and when one of these is mutated, the new value is not freely chosen, only a small random change in the current value is allowed. These limitations results in smaller changes in the genotype, and are implemented to make it easier to guide the evolution in correct direction. The node where the mutation is to be performed is selected at random. Figure 8.4 shows an example of a mutation operation.

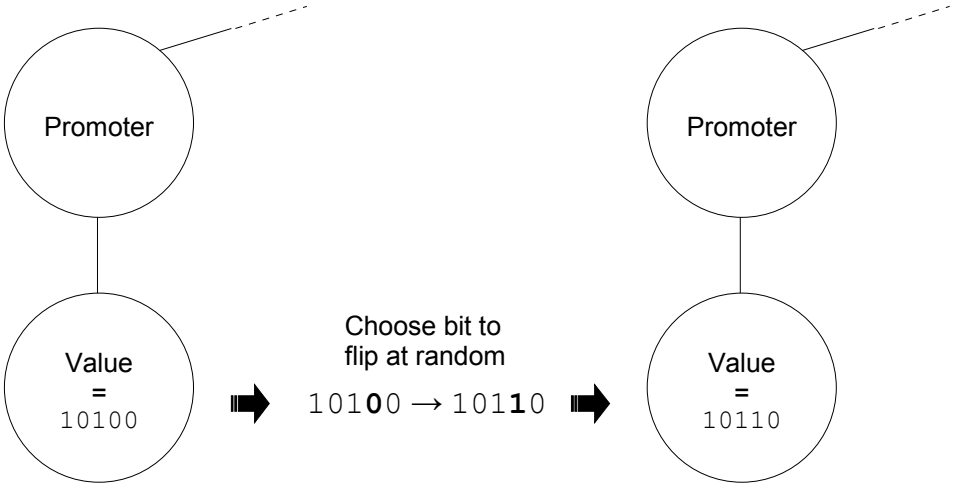


Figure 8.4: An example of the mutation operator applied at the node level. The value of the promotor is changed from 10100 to 10110 (bit number four is flipped)

The mutation operator working on the top level of the genotype, is special. It does not mutate in the classical sense (changing the content of a node), but instead introduces or removes a gene. The introduction of a gene is accomplished through gene duplication, a mechanism known from biology. This mechanism has also been used with some success in artificial development systems [BONG01]. Gene duplication may be useful because although the duplicate gene may not result in any change in the phenotype right away, it may do so after further mutations (at the node level). This should make it possible to do smaller and more gradual refinements of the phenotype. Gene removal is useful for deleting excess gene (genes not contributing to the development) and disruptive genes (a more fit phenotype would be developed without these genes). Genes which are contributing to the development may also be accidentally deleted by this mutation, but as these "good" genes will have already been spread throughout the population, deleting a few of them should not be too harmful. For an example of how gene duplication and gene removal works, see figure 8.5 and 8.6, respectively.

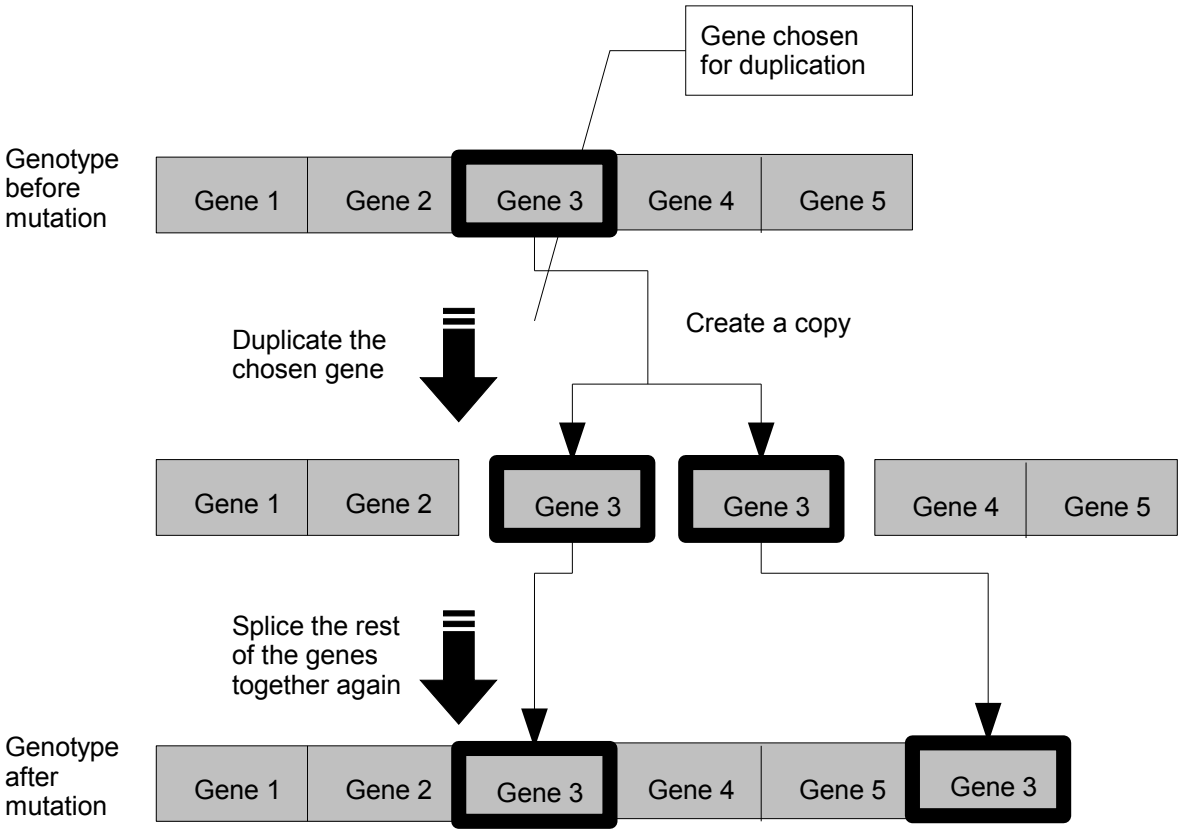


Figure 8.5: The mutation operator applied at the top level of the genotype. Shows an example of gene duplication, where gene 3 is selected for duplication, and the duplicate is inserted at the end of the gene list

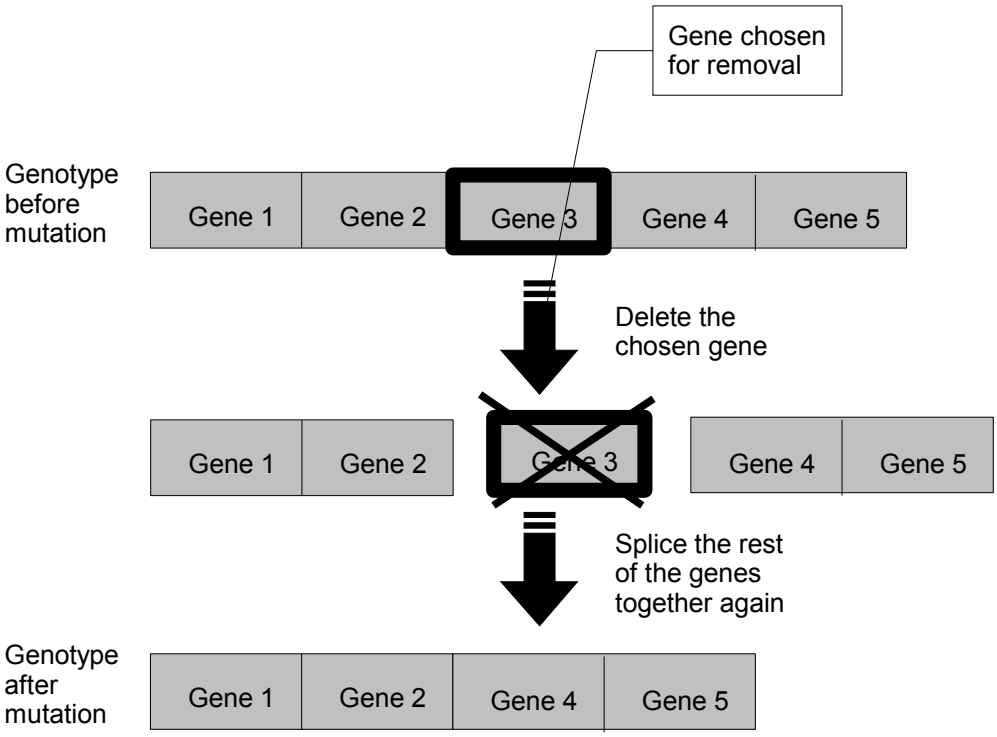


Figure 8.6: The mutation operator applied at the top level of the genotype. Shows an example of gene removal, where gene 3 is selected for removal.

8.2.4 Fitness Function

ArtDev3D is designed to support various fitness functions. Currently, only one fitness function is implemented: CellByCellComparisonFitness. This function compares the phenotype to evaluate against the target, and returns a normalized fitness – see chapter 3.2.4.

The algorithm used when comparing the two phenotypes is as follows:

1. Compare each location in the phenotype to evaluate against the corresponding location in the target phenotype.

    - If both locations are empty (no cell) add two to a temporary fitness sum.
    - If both locations are occupied by a cell, and the cells are of the same type, add two to a temporary fitness sum.
    - If both locations are occupied by a cell, and the cells are of different type, add one to a temporary fitness sum.
    - If one location is occupied by a cell while the other is not, the temporary fitness sum is not changed.

2. Calculate the maximum reachable value for the temporary fitness sum (the number of locations in the target phenotype times two)

3. Divide the temporary fitness sum with the maximum reachable sum to find the normalized fitness.

8.2.5 Selection Mechanism

The tournament selection algorithm was chosen as the selection mechanism to use in ArtDev3D. In tournament selection, a subset of the population, the tournament group, is selected at random and a selective competition between the individuals in this group takes place.

Tournament selection was chosen for two reasons: it does not require that the fitness of all individuals in the population is evaluated, and the algorithm has proven itself to handle premature convergence better than the alternative selection mechanisms. The former reason is especially important in ArtDev3D because the development from genotype to phenotype is such a time-consuming process. The latter reason touches on a critical problem with GPs: as evolution goes by, the population tend to loose its diversity, resulting in premature convergence in the population. This happens when the GP gets stuck in a local optimum, making it unable to find the global optimum.

8.3 The Artificial Development Process

The development part of the system can be run in one of two ways: either as a stand alone application, or in combination with an evolutionary algorithm. Either way, the development process takes a DNA, the number of development steps to use, and the bounds of the organism to develop as inputs. The output of the process is the full-grown organism. An overview is given in figure 8.7.

Figure 8.7: The development process

The interesting part here is what happens during the development process. But before that, a summary of the different components in this model is necessary.

8.3.1 The Components

The major components used in this implementation are much the same as the ones used in BioDev – see chapter 7.1: the organism, cells, the DNA, proteins and chemicals. Each will now be described.

The Organism

The organism is a three dimensional grid with a fixed size. A location inside the organism may be either empty or filled by a cell. Each cell may occupy exactly one location within the grid, and only one cell is allowed at each location.

The biggest change from BioDev is that the organism now is three dimensional. A more subtle difference is that the notion of universe does not exist any more. It has been replaced by a broader definition of the term organism, which now also encompasses what was previously called the universe.

The Cell

The cells have a fixed size; they occupy exactly one location within the organism grid. Inside each cell is a DNA and (possibly) a number of proteins and chemicals. Each cell is of a specific type, which is one of a set of explicitly defined cell types.

The cells are capable of performing a variety of actions:

- Cell division – the cell divides, creating copies of itself which is placed in the neighbouring location in one or more directions
- Change type – the cell changes its type it is
- Transcribe genes – transcribes one or more genes
- Produce / consume chemicals – produces or consumes some amount of one or more of the chemical types inside the cell

A number of changes have been done on the cell from BioDev. While in BioDev, the type of the cell was implicitly defined through differences in gene expression patterns, in ArtDev3D, the type of the cell is explicitly defined. The cell may only be one of a limited set of cell types. The available actions also have changed. The cell may no longer kill itself, move or send and receive signals. Although these actions are biologically plausible, they complicates the development process unnecessarily (as experienced with BioDev), and are hence not included in ArtDev3D. A final change is the removal of the cell's precondition, which was removed for much the same reasons as the actions.

The DNA

The DNA consists of a number of genes. Each gene is composed of two parts; the promotor and the coding region. The promotor is the gene marker, which is used when deciding which

genes are to be transcribed. The coding region contains all the information necessary to construct the protein the gene codes for.

In contrast to BioDev, it is in this model not possible to block a gene from transcription by putting an inhibitor in the regulatory region. This choice was made because the blocking of particular genes is thought to be a complexity that is not necessary with respect to the goal of ArtDev3D.

The Protein

The protein has a specific time to live, a function it performs, and a precondition which must be fulfilled in order for it to carry out its function.

The time to live specifies how many clock ticks the protein is active, before it dies and is removed from the cell.

A protein can have one of the following functions:

- Divide cell – request the cell to perform a division
- Change chemical concentration – request the cell to produce or consume chemicals
- Change type of cell – request the cell to change its type
- Transcribe genes – request the cell to transcribe genes

A protein is not able to directly perform any actions, it can only request the cell it resides in to perform the action for it. If and how the action is actually performed, is decided by the cell.

The precondition of the protein consists of two parts. One part checks the neighbourhood surrounding the cell, while the other part checks the concentration of chemicals in it. The neighbourhood part of the precondition checks each of the neighbouring locations in the directions up, down, left, right, in front of and behind for the presence of a cell and, if present, the type of cell. This part of the precondition is fulfilled if all neighbours fits the specified neighbourhood pattern. The chemical part of the precondition checks the concentration of each of the chemical types in the cell to see if it is above the specified level. If the concentrations of all the chemical types are above their specified level, this part of the precondition is fulfilled.

One of the most significant differences from BioDev regarding the protein, is the way proteins affect its environment. In BioDev, proteins where allowed to perform their function directly, and in a first-come-first-serve fashion. In ArtDev3D, proteins are not allowed to affect anything directly. They are only allowed to request the cell to perform actions for them, and the cell does not make the decision of whether to perform the action or not until all the proteins in the cell have performed their request for the current tick. This way of handling the protein actions eliminates the problem of deciding in which order the proteins should be allowed to perform their actions. Because of this, a kind of parallelism is achieved at the cellular level.

In nature, the complex structuring of the protein decides the function of the protein – see chapter 4.1.2. In ArtDev3D this is greatly simplified in the way that the protein has an explicit type and function, instead of implicitly specified by its shape.

Chemicals

Each cell may contain zero, one, or more chemical types. It is not which chemical types are present in a cell that is important, it is the concentration of each type that is important. The concentration of a chemical type may change with time. All the cells contains the same number of chemical types, and this number must be decided before running the development process.

The use of chemicals is a novel feature in ArtDev3D, not implemented in BioDev. Chemicals where included because they are thought to help when developing more complex organism [MILL03].

8.3.2 Overview of the Development process

The development process is accomplished in three steps: initialization, development, finish.

Initialization

First, an empty cell is created. The DNA given as input is put inside this cell, and all the genes in the DNA are transcribed. Then, an organism is created using the given bounds, and the cell is placed in the middle of this organism – see figure 8.8. The development process is now ready to run.

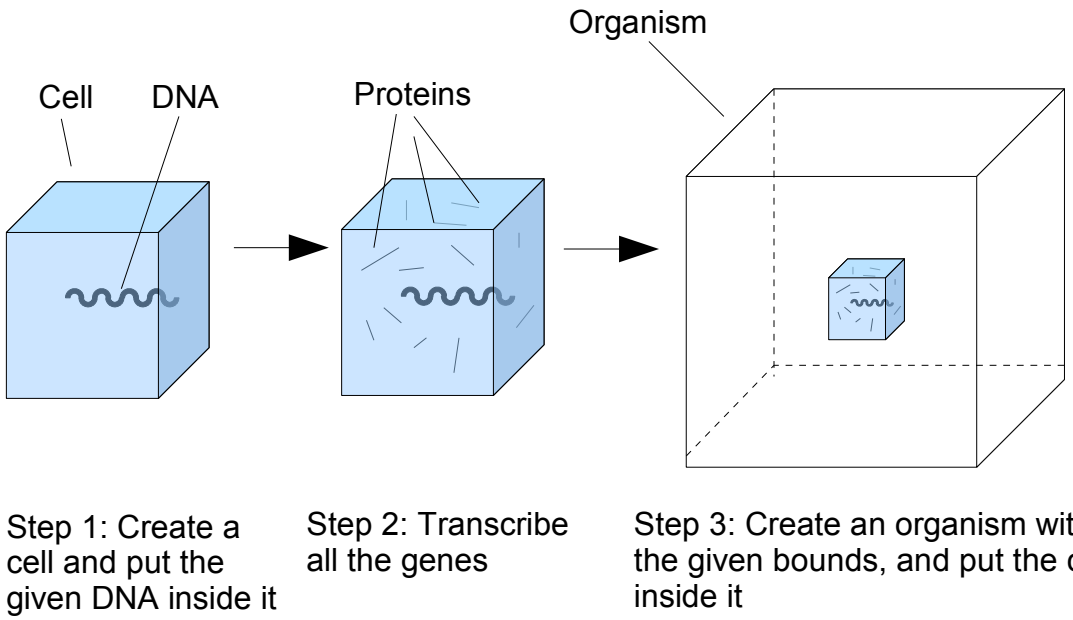


Figure 8.8: Initialization of the development process

Development

The process of development is divided into discrete time steps called ticks. For each tick, all the cells currently in the organism are allowed to perform their actions, like cell division and protein production. Unlike in nature, the cells perform their actions one after the other, and not in parallel. This has serious implications for the result of the development process, but is necessary to keep the model simple enough. An example of a developing organism is given in figure 8.9.

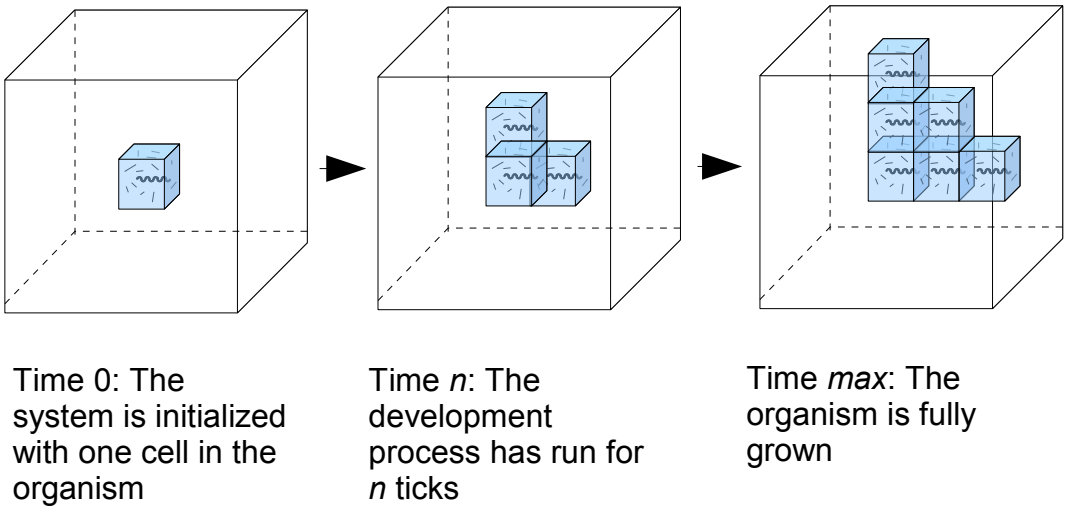


Figure 8.9: The development of an organism

Finish

The development continues until the organism has developed for the given number of ticks. The full-grown organism is then given as output of the system.

8.3.3 Details of the Development Process

The development process can be separated into two levels of detail: at the organism level, and at the cell level. A description of the process at the level of the organism will be given first, followed by a description at the level of the cells.

The Organism Level

Each time the development process proceeds one step, the organism is notified that a tick has happened. The organism, consisting of cells and empty space, notifies each of its cells that a tick has occurred. The cells are notified one after the other, so the actions a cell performs may be affected by the actions of the cells receiving the tick notification before it. The cells are not notified of the tick at random; they are notified in the order in which they were created. This means the zygote is notified first, then the first cell created by the zygote, and then it continues in this fashion until the last created cell is notified.

Most of the actions that a cell can perform are affected indirectly through the neighbourhood part of the protein precondition. This is because if a cell divides and places its daughter cell in the neighbourhood of another cell, which is notified of the tick after this cell, the other cell's neighbourhood has changed. Hence, other proteins will perform their functions than if the two cells were notified of the tick in opposite order.

An example of this is given in figure 8.10. The organism is here represented as a two dimensional grid instead of a three dimensional one. This is done to make it easier to see what is happening. At time *n* (figure A) the blue cell wants to divide downwards and the green cell wants to change it type to yellow. However, the protein requesting the change in cell type has a neighbourhood precondition which says that all neighbours must be empty. If not, it will not request the cell to change its type. How the organism looks at time *n+1* depends on which one of the two cells are notified of the tick first. In figure B, the green cell received the notification first, and hence was able to change its type to yellow. After this, the blue cell are allowed to divide. In figure C, the blue cell is notified first. The blue cell divides downwards, but when the green cell is notified of the tick, its neighbour in direction up is a cell and hence the protein responsible for the change of type will not request the cell to change type.

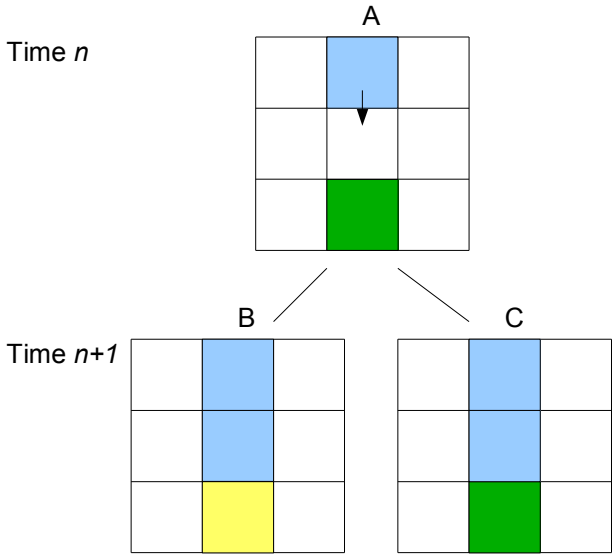


Figure 8.10: In A, the green cell wants to change its type, the blue cell wants to divide downwards. Because of the neighbourhood precondition of the protein requesting the change in cell type, the organism may look like either B or C after one tick, depending on which cell is notified first of the tick

Because ArtDev3D implements the mechanism of contact inhibition, the divide cell action is directly influenced by the actions of other cells, as a cell may only divide in a given direction if the neighbour location in that direction is empty.

An example of this is given in figure 8.11. As in figure 8.10, the organism is also here represented as a two dimensional grid. This time, a blue cell and a green both want to divide and place their daughter cell at the same location. This is the situation at time *n* (figure A). At time *n+1*, the organism will be either figure B or figure C, depending on which of the two cell are first notified of the tick. If the green cell is first notified, the organism will be figure B. If the blue cell is first notified, the organism will be figure C.

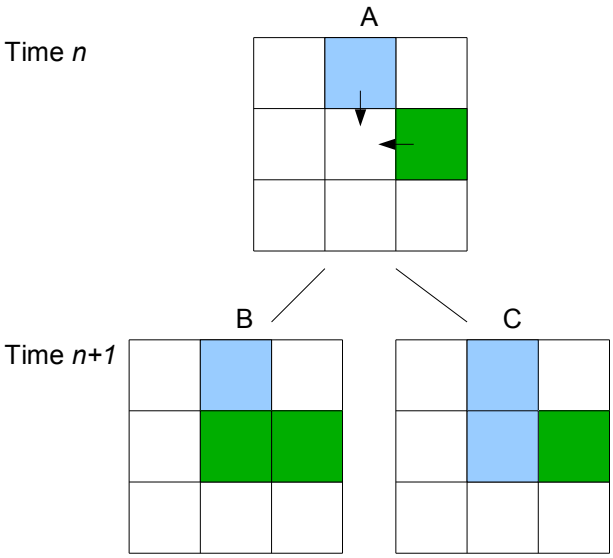


Figure 8.11: Both cells want to divide into the same location in A. The result after a tick has passed will be either B or C, depending on which of the two cells is first notified of the tick

The Cell Level

Each time a cell is notified of a tick, the same series of discrete steps occur: the chemical concentrations are normalized, the proteins are notified of the tick, and the cell performs the actions requested by the proteins.

The normalization of the chemical concentrations is simply that if they are out of their predefined bounds, they are set to the nearest legal value. In other words, if a chemical concentration above the specified max value, it is set to the max value, and it is set to the min value if below min.

As proteins cannot change their environment in such a way that it affects the actions of the other proteins, the order in which they are notified of the tick is unimportant. As discussed above, this way of handling the proteins is a way to achieve some of the massive parallelism occurring in biological processes. When notified of the tick, each protein with fulfilled preconditions performs its request to the cell. Dead proteins are collected and removed from the cell.

When all this is completed, it is the cell's turn to do the requested actions. As the different types of actions cannot be performed in parallel, some sort of ordering has to be made. A choice has been made to perform the actions in the following order: transcribe genes, change chemical concentrations, divide cell, and change the cell's type. How each action is performed differs to some extent, and each will now be described.

Transcribe Genes

All requested transcriptions are performed.

Change Chemical Concentrations

Changes all the chemical concentrations as requested by the proteins. The concentrations are allowed to exceed the upper and lower bounds during this process. This is done to avoid the ordering of the request to having any influence on the final concentrations.

Divide Cell

The cell is able to divide once in each of the six directions during one tick. Whether the cell divides in a direction is decided by the value of the corresponding stimuli level, making a total of six stimuli levels for this action. The stimuli levels are reset each tick. Each request made by the proteins affects all these six stimuli levels, either by increasing or decreasing each of them. A request may also have a neutral effect on one or more of the stimuli levels. When all the requests have been processed, each stimuli level is checked against a threshold value and, if above this threshold, the cell decides to divide in the corresponding direction. If the cell actually divides in the directions it decides to divide depends, for each direction, whether the location to divide into is empty and within the bounds of the organism. If so, the cell will divide.

Change the Cell's Type

As with the divide cell action, this action also uses stimuli levels to make a decision, one level for each cell type. First, all the requests are processed, changing the stimuli level of choice. Then a winner is chosen (the highest stimuli level) and, if this stimuli level is above the specified threshold value, the cell will change its type to the corresponding one.

9 Testing

When designing the ArtDev3D system, three features of development (as it works in nature) were deemed important to achieve:

- Cell division – necessary for the organism to grow in size
- Cell differentiation – allows for specialisation of cells
- Morphogenesis – the shaping of the organism

Cell division is supposed to be achieved through the protein DivideCellProtein, and cell differentiation through ChangeTypeOfCellProtein. Morphogenesis on the other hand, is not achieved through a specific protein, but rather through the interdependencies and interactions of a collection of proteins. Or, in other words, morphogenesis is supposed to be achieved through the gene regulatory system.

9.1 Issues

In practice there are a number of concerns regarding the development process: as proteins only can be transcribed (“created”) by another protein, namely TranscribeGeneProtein, how should the whole process of gene transcription be initiated? Initiating the cell with no proteins at all is out of the question as that would make it impossible to achieve both cell division, cell differentiation and morphogenesis. What is necessary to achieve any of these three features is the presence of at least one TranscribeGeneProtein in the cell when it is initiated. Exactly how the presence of such a protein is made certain, is another important question. In this series of testings, the cell is initialized in the following way: for each gene in the cell's DNA, create the corresponding protein. This does not guarantee that a TranscribeGeneProtein is present in the cell, but it seems reasonable to believe there is a fair chance one TranscribeGeneProtein capable of igniting the gene regulatory system of the cell will be made present using this method.

Other concerns are the length of a genes promotor and the length of the sub promotor used by TranscribeGeneProtein to find genes to transcribe, the discrete time to live for a protein once transcribed, the number of genes in the DNA, and the time to allow the organism to develop before it is considered full-grown.

Through some informal test runs, usable values for these parameters have been found. The values found may not be the optimal ones, but they have proven themselves to be good enough for initial testing.

9.2 Initial Number of Don't Care Neighbours

Each protein used in ArtDev3D has a precondition which, if fulfilled, activates the protein. The precondition has a check for both the neighbouring cells and for the concentration of chemicals within the cell. Initial number of don't care neighbours (IDN) concerns the former part of the precondition. It checks for both the presence/absence of adjacent cells and their cell type. The

cells checked are the ones to the right, left, above, below, behind and in front of the cell where the protein resides. The neighbourhood part of the precondition has six values, one for each direction. The allowed values are: cell type, no cell, or don't care (wild card). All values must match with the cell's neighbourhood for this part of the precondition to be fulfilled.

Because a more specifically defined neighbourhood in the precondition allows for more specificity when growing an organism i.e. allows more complex organisms to be grown, using as few don't cares as possible may be desired. However, a more specific neighbourhood also means that it will be more difficult to start the development (i.e. it will be difficult to achieve any growth). With little or no growth it will be very hard for the genetic algorithm to search in the correct direction, resulting in poor fitness score. Because of this the number of don't cares may need to be relatively high.

IDN is used as a term to describe the number of don't cares the genetic algorithm puts in the neighbourhood part of a protein's precondition when creating the first random generation. The number of don't care neighbours may well change during the course of evolution: IDN only denotes the starting condition.

9.3 Number of Chemical Types

As already discussed, the precondition of a protein has two parts: one checking the neighbourhood and one checking the chemical concentration. While IDN concerns the former of these two, number of chemical types (CHT) concerns the latter. This part checks to see whether the concentrations of the different chemicals are above their specified thresholds. If all chemical concentrations are above, this part of the precondition is fulfilled.

Increasing the number of chemical types may be a way to achieve the growth of more complex organisms. This is because, like decreasing IDN, increasing CHT makes the precondition more specific, possibly allowing for more detailed structuring of the growth process.

9.4 The Experiments

All the three desired features of the development process were tested for. As cell division is required for both of the two other features to appear, this feature is first tested alone. To be able to test for any of the two other features, the system must first be able to achieve cell division and an acceptable growth of the organism.

9.4.1 The Growth Experiments

The experiments performed regarding the cell division feature are referred to as the growth experiments. Because of the uncertainty regarding what the best value is for some of the parameters, the growth experiment series was run with a collection of parameter values. The two connected with the most uncertainty and whose effects on the development process are unknown, are the number of chemical types (CHT) to use and the specificity of the neighbourhood part of the proteins precondition: the initial number of don't care neighbours (IDN). Seven different values were chosen for each of these two parameters (zero to five plus

ten for chemicals, zero to six for neighbours) and experiments were run for each possible combination of these, resulting in a total of 49 experiments.

The hypothesis behind the growth experiments is that a shape with fewer dimensions will be easier to develop. Therefore, three different target shapes were created: a one dimensional stick, a two dimensional plane, and a three dimensional cube – see figure 9.1, 9.2 and 9.3 respectively. The 49 experiments discussed above where performed for each of these targets, resulting in a total of 147 experiments for the growth experiments.

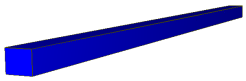


Figure 9.1: One dimensional target phenotype

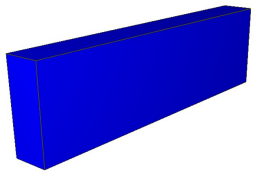


Figure 9.2: Two dimensional target phenotype

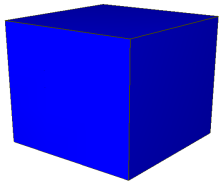


Figure 9.3: Three dimensional target phenotype

Only simple morphogenesis is necessary when developing these targets, as the whole interior i.e everything within the bounds of the organism, is filled with cells. This means that the growth of the organism need not be restricted by the DNA to achieve the correct shape, as the shape will be formed by the bounds of the organism (cells are not allowed to divide outside the bounds of the organism).

The results from these experiments showed that the system did not find it particularly difficult to evolve any of these three phenotypes: roughly half the runs achieved perfect fitness, while the remaining half achieved close-to perfect fitness. The three dimensional phenotype achieved, on average, higher fitness compared to the other two phenotypes, followed by the two dimensional one, and then the one dimensional one. Although far from statistically significant (because of the small number of runs performed), this trend seems highly likely for the phenotypes used here. Increasing the number of dimensions also means an increase in compactness. The three dimensional phenotype may be perfectly developed in only two steps, requiring one single protein (DivideCellProtein which divides twice in all direction) to achieve this. The one dimensional one, on the other hand, requires at least 13 steps to be perfectly developed, and in addition requires the DivideCellProtein to be transcribed (activated) multiple times.

There seemed to be an increase in fitness with increasing IDN, and a decrease with increasing CHT. However, for IDN values of four to six and CHT values of zero to two, there seemed to be no real difference in the achieved fitness.

9.4.2 The Morphogenesis Experiments

It was now established that ArtDev3D is capable of growing an organism, but achieving cell growth alone is not enough. A more detailed control over the shaping of the organism than the bounds can provide, is desired.

There are several ways to achieve morphogenesis. The way morphogenesis is implemented in ArtDev3D is that cell division (growth) may cease to continue at specific locations, thereby restricting the growth of the organism. Other ways to achieve morphogenesis is through cell movement, cell shaping and apoptosis (cell death), but these have not been implemented. To test ArtDev3D's ability to restrict the growth of the organism three new targets were designed. These three targets have a border of empty space between the outermost cells and the bounds of the organism, requiring the cells to stop dividing before the bounds – see figure 9.4, 9.5 and 9.6. The setting and parameters used when running this experiment series are the same as the ones used in the growth experiment.

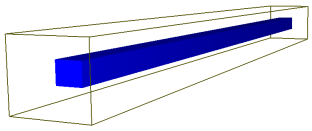


Figure 9.4: One dimensional target phenotype. The lines represent the bounds.

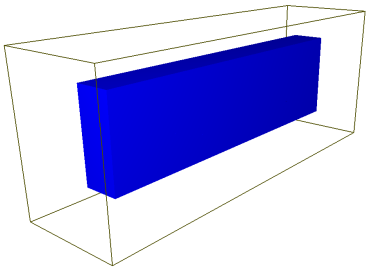


Figure 9.5: Two dimensional target phenotype. The lines represent the bounds.

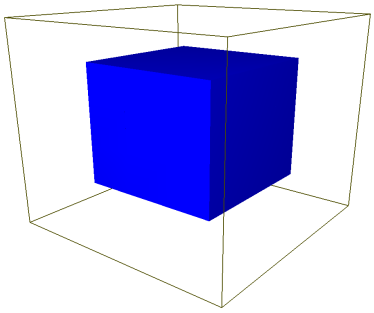


Figure 9.6: Three dimensional target phenotype. The lines represent the bounds.

Developing phenotypes with shapes requiring morphogenesis, as expected, seems to be more difficult. Although perfect fit organisms for both the one dimensional and the two dimensional cases were found, no perfect fit was found for the three dimensional one. However, fairly high fitness (0,97) was achieved in a number of runs and the variation in achieved fitness was small. The two dimensional phenotype on average achieved better fitness than the two other, followed by the one dimensional one, and finally the three dimensional one.

There was no obvious difference found in the achieved fitness with the various values of IDN and CHT.

As the experiments performed were run with both a relatively small population size and few generations, it is believed that increasing the population size and / or the number of generations will result in better fit individuals. If so, also the runs using the three dimensional target should be able to evolve a perfect fit individual. One reason why no perfect fit was found when using the three dimensional target may be because this target requires first growth

and then a stop in growth in six direction, while the two dimensional an one dimensional only requires this in four and two directions respectively.

These experiments showed that ArtDev3D is capable not only to grow a random blob of cells, but also stop the growth process at desired location in order to shape the growing organism.

9.4.3 The Differentiation Experiments

This experiment series takes us one step back, forgetting all about the previous morphogenesis experiment and builds solely on the growth experiment. What was tested now, was the systems ability to differentiate cells.

These experiments use the same parameters and phenotypes as in the growth experiment, described above. The only difference is the introduction of a second cell type in each of the three phenotypes – see figure 9.7, 9.8 and 9.9. A cell type is recognised by its unique colour.

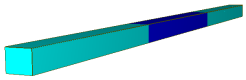


Figure 9.7: One dimensional target phenotype.

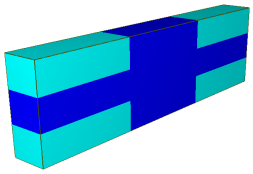


Figure 9.8: Two dimensional target phenotype.

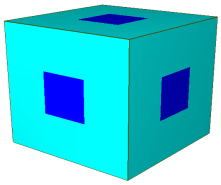


Figure 9.9: Three dimensional target phenotype.

As with the growth experiments, no morphogenesis is necessary when developing these targets, as the whole interior i.e. everything within the bounds of the organism, is filled with cells.

To achieve the patterns present in each of the three phenotypes, some sort of differentiation has to happen at some point in the development. This challenge seems to have been a hard one for ArtDev3D. The system was only able to develop an organism with perfect fitness in one of the runs for the three dimensional target phenotype. However, the purpose of these experiments were simply to test if the system was capable of performing differentiation, and this was accomplished for all the three phenotypes.

9.4.4 The Morphogenesis Combined With Differentiation Experiments

So far it was shown that ArtDev3D is indeed capable of performing all the three features. It was, however, also interesting to test if the system is capable of handling both morphogenesis and differentiation at the same time. This is the purpose of this final experiment series in the testing phase.

Once again, the parameters and settings were the same as the ones used in the growth experiment, described above. The shape and colour pattern are the same as in the differentiation experiments, while the border of spaces are the same as was used in the morphogenesis experiments – see figure 9.10, 9.11 and 9.12.

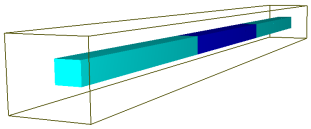


Figure 9.10: One dimensional target phenotype. The lines represent the bounds.

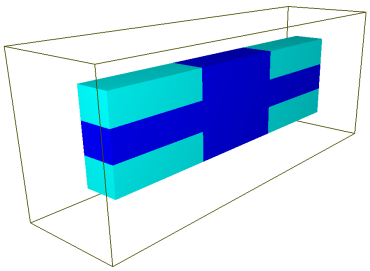


Figure 9.11: Two dimensional target phenotype. The lines represent the bounds.

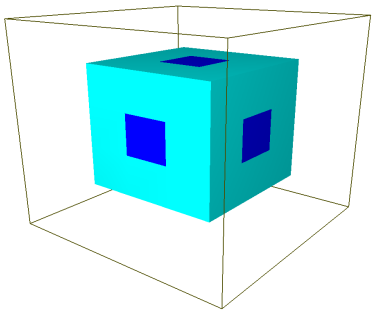


Figure 9.12: Three dimensional target phenotype. The lines represent the bounds.

The results showed that ArtDev3D was unable to find a perfect fit for any of the three phenotypes used as targets. This is not as surprising, keeping in mind the results from both the morphogenesis and the differentiation experiments. There is no reason to believe this experiment series should be any easier for the system.

More surprisingly, the fitness achieved in these experiments is generally higher than the fitness achieved in both the morphogenesis and the differentiation experiments. One reasonable explanation for this is the way the fitness function calculates the fitness: it is easier to achieve higher fitness when there are lot of empty space between the outer cells and the organism bounds. This is because the empty border makes up a large percentage of the available locations in the organism, so the fitness is automatically relatively high as long as the organism does not develop into these locations.

9.5 Conclusion

This series of experiments has shown that the ArtDev3D system is capable of performing the three desired features cell division, cell differentiation, and morphogenesis. It has also been established that the chosen values for the system parameters in these experiments are

reasonably good This is implied by the high fitness achieved despite the constraints in the genetic algorithm (only 50 generations and a population size of 100). Another limitation probably originating from the genetic algorithm is the fact that the fitness for the population increases fast during the first few generations and then stagnates. This is an indication that some of the parameters used for the genetic algorithm will require some tweaking.

Although relatively high fitness was achieved in most of the runs, the fact that the system was unable to develop a number of the targets perfectly indicates that ArtDev3D has its limitations. The targets in question are relatively small, meaning the solution space is also small, so the search should not have been that hard. The fact that the search was indeed difficult, is an indication that maybe the target is not possible to express using the available components, or that maybe the guiding of evolution, as provided by the fitness function, is not good enough.

One of the hypothesis made before the experiments were run was that a one dimensional phenotype would be easier to evolve than a two dimensional one, and the three dimensional phenotype would be the hardest to develop. No evidence supporting this hypothesis was found during these experiments, so this hypothesis had to be rejected.

The results from these experiments has shown that the system is capable of developing organisms requiring both morphogenesis and differentiation. The relatively high fitness achieved despite the sub optimal settings in the genetic algorithm, is promising for the experiments to come.

Part III – Experiments

10 Experiment 1

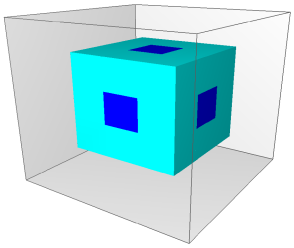
The initial testing of the ArtDev3D system showed that the system was capable of developing simple three dimensional shapes. However, for very simple three dimensional shapes the biological mechanism of cell division is sufficient, as shown in chapter 9. The experiments went further to show that the model could also handle cell differentiation and morphogenesis, mechanisms required for more advanced three dimensional shapes. It was now time to, not only explore what types of three dimensional shapes the model is capable of developing, but to investigate the model itself. A fuller understanding of the effect of the model's parameters and their inter-relationship was needed so as to be able to refine the model.

10.1 Setup

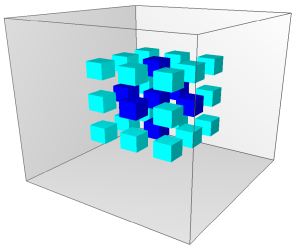
Five phenotypes were designed to use as targets for this experiment series: a cube embedded with a three dimensional cross inside, a tree, a christmas tree, a sphere, and a sphere which is separated in the middle. See figure 10.1 for an overview of these phenotypes. Each phenotype is shown in two different ways: normal view and exploded view. The normal view is how the phenotype actually looks like, while the exploded view shows the phenotype with space around each cell. The exploded view is provided to get a look “inside” of the three dimensional phenotype.

Three different shapes were used for these phenotypes: the cube, the tree, and the sphere. These shapes were chosen to represent classes of shapes which provide different challenges for development. The cube is a regular target, requiring the same growth and differentiation in all six directions, the tree is a bit more complex, requiring different growth in several directions, and the sphere is fairly simple, possible to achieve just by having the cells to divide in all six directions two times. What distinguishes the tree from the christmas tree, and the sphere from the divided sphere, is the number of different cell types needed when designing them. ArtDev3D allows for a number of different types of cells in the organism, each distinguished by its unique colour. The use of different number of cell types is done to test for differences in the results when using more or less cell types, keeping the shape constant.

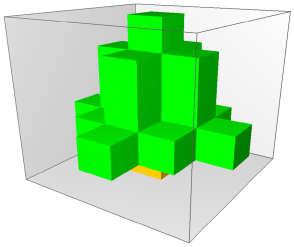
Numerous different values for the same two parameters investigated in chapter 9, the number of chemical types (CHT) – see chapter 9.3, and the initial number of don't care neighbours (IDN) – see chapter 9.2, were used when running experiments for each of the target phenotypes. IDN denotes the number of don't cares in the neighbourhood part of each protein's precondition in the very first generation created by the genetic programming algorithm. For CHT the set of values used was zero up to five and ten, while for IDN all allowed values (zero up to and including six) were used. For each phenotype all possible combinations of these two sets of values were conducted and the experiment run for each combination was repeated twenty times. This was done both because of statistical relevance, and to make sure the achievement of developing a shape was not just by chance to ensure the system has a certain degree of stability.



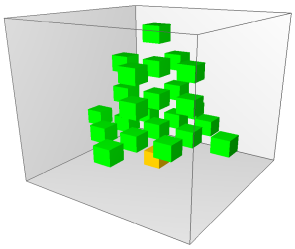
Cube – normal view



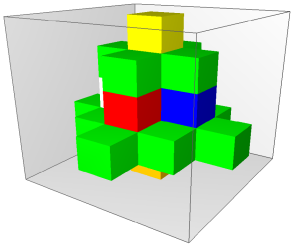
Cube – exploded view



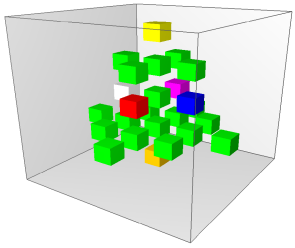
Tree – normal view



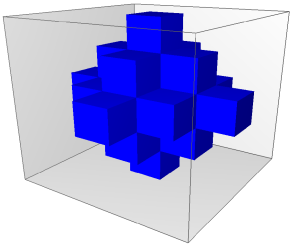
Tree – exploded view



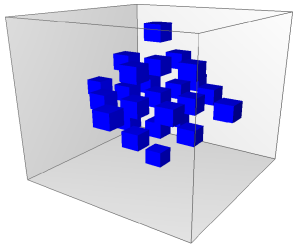
Christmas tree – normal view



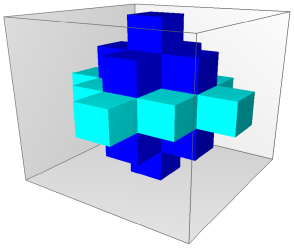
Christmas tree – exploded view



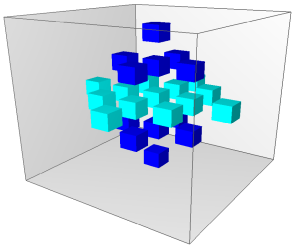
Sphere – normal view



Sphere – exploded view



Divided sphere – normal view



Divided sphere – exploded view

Figure 10.1: The five target phenotypes

10.1.1 Parameters

Most of the values used for the parameters are the same as during the testing phase – see chapter 9. However, due to the results of these experiments it was chosen to increase the population size and the number of generations. The size of the population is increased from 100 to 1000 and the number of generations is increased from 50 to 500.

The parameters were held constant during all five experiments except for the number of cell types to use, the number of chemical types (CHT), and the initial number of don't care neighbours (IDN). For the two experiments using the tree and the christmas tree, more cell types where allowed to be chosen among than what was absolutely necessary. This was done so that "natural" colours could be used on the trees (like green on the leaves) without having to make any changes in the system's program code. The cell types are implemented in such a way that it is not possible to specifically chose which cell types to make available to the system: if one cell type is needed, cell type one is made available to the system, if two cell types are needed, cell types one and two are made available, and so on.

Genetic Algorithm

The following are the parameters used by the genetic algorithm.

General settings

Population size = 1000
Crossover rate = 0.9
Mutation rate = 0.1
Termination criterion = Generations=500 OR fitness=1.0
Fitness function = *CellByCellComparisonFitness*
Selection method = *TournamentSelection*

- Group size = 4

Genotype specification

Promotor = 6 bits
Initial no genes = 5
Proteins used = *GAdivideCellProtein*
GatranscribeGeneProtein

- Length of subpromotor = 3 bits

GachangeTypeOfCell

- Cube – number of allowed cell types: 2
- Tree – number of allowed cell types: 8
- Christmas tree – number of allowed cell types: 12
- Sphere – number of allowed cell types: 1
- Divided sphere – number of allowed cell types: 2

Proteins
Max time to live = 5 ticks
Precondition = Chemicals – number of different types = {0, 1, 2, 3, 4, 5, 10}

Neighbours – initial number of don't cares = {0, 1, 2, 3, 4, 5, 6}

Target phenotypes

Five different phenotypes are used as targets:

1. Cube – number of allowed cell types: 2
2. Tree – number of allowed cell types: 8
3. Christmas tree – number of allowed cell types: 12
4. Sphere – number of allowed cell types: 1
5. Divided sphere – number of allowed cell types: 2

All the phenotypes have the same dimensions: five-by-five-by-five units.

Development

Development time is the number of steps (ticks) the organism is allowed to live before it is evaluated by the fitness function. All the experiments in this series use the same development time of 12 ticks.

The number of ticks to use for development is calculated using the following formula:
$x = (a + 5) * 1,5$

Where:
x is the calculated development time
a is the minimum needed ticks to grow to wanted size for the biggest of the targets (calculated by hand)
5 is added to ensure enough time for differentiation to happen
1,5 is multiplied to this sum to give extra time to develop

10.2 Running the experiments

All the experiments in this series were run on the two Beowulf clusters Clustis2 (clustis.idi.ntnu.no) and Norgrid (norgrid.ntnu.no). Clustis2 and Norgrid features 22 and 63 computational nodes respectively. Each node contains 1GB of memory and a 3.4GHz Pentium IV processor. The two clusters use Linux as the operating system, and Torque as the program for queuing the jobs to be run .

10.3 Results

A significant amount of data is generated with so many development experiments. It was, therefore, important to select which data should be stored for each experiment, providing sufficient information to enable the analysis sought to be undertaken. The following data was used for this analysis: the fitness of the best evolved individual in each repetition of each single experiment, along with the DNA used to construct it, and its phenotype representation.

The final fitness is used to plot graphs to visualize the “goodness” of the various runs, and the stability of the results. The stability of the results refers to the systems ability to repeatedly

evolve equally fit individuals in separate runs using the same parameters and target phenotype. The standard deviation, which is plotted on top of the fitness bars, is a measure for this stability.

Two graphs was plotted for each of the targets: one showing the fitness with respect to CHT, and one showing the fitness with respect to IDN. For each CHT value, the average fitness over all runs performed using this CHT i.e. all runs with a combination of this CHT value and any of the IDN values, was used. Similarly for the IDN values: here the average fitness over all the CHT values was used. Also the standard deviations were calculated in this manner.

The DNA of the best evolved individual, regardless of CHT and IDN, for each target phenotype was used to investigate the development process the individual goes through, from the zygote to the full-grown organism.

The phenotype of the best evolved individual, regardless of CHT and IDN, for each target phenotype was used to investigate the differences to the corresponding target phenotype.

The fitness results for each of the target phenotypes will be discussed next, followed by a brief discussion of the step-by-step development the best evolved individual goes through.

10.3.1 Cube

Nine out of the 980 experiment runs where the cube was used as the target phenotype achieved perfect fitness. The perfect fitness was typically achieved in runs with high IDN, combined with low CHT. This impression is strengthened when analysing the graphs plotting the fitness data with respect to CHT and IDN – see figure 10.2 and 10.3, respectively. The tendency is for low values of CHT, and higher values of IDN, resulting in high fitness.

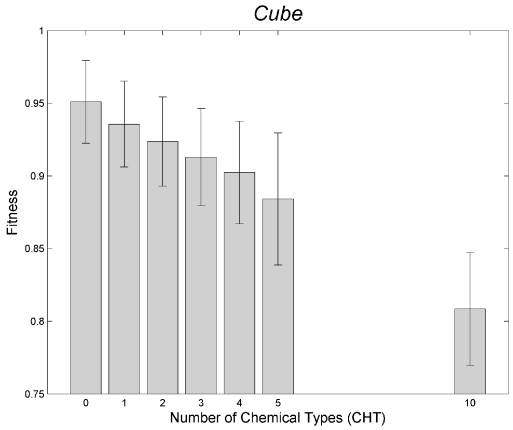


Figure 10.2: Fitness achieved when developing the cube with different numbers of chemicals (CHT)

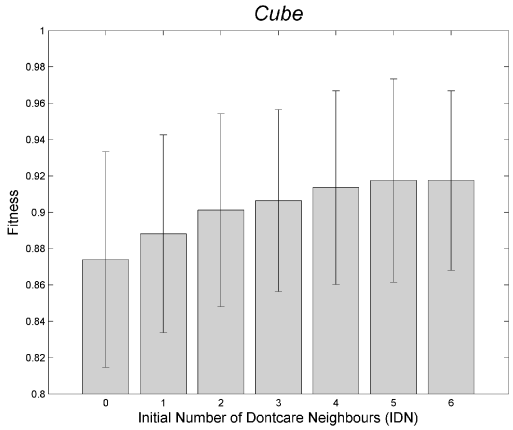


Figure 10.3: Fitness achieved when developing the cube with different initial number of don't care neighbours (IDN)

The standard deviations are roughly equal large regardless of CHT and IDN. This means that the achieved fitness do not become more or less random when these two values are changed, indicating that with increasing CHT, or decreasing IDN, the achieved fitness goes steadily down without inducing any larger changes in the systems ability to guide evolution.

10.3.2 Tree

None of the experiments using the tree as target phenotype were able to achieve perfect fitness. However, four of the runs achieved a fitness of 0,996. As in the cube experiments, this high fitness was achieved in runs using low CHT values and high IDN values. Looking at the graph displaying the fitness data with respect to CHT and IDN – see figure 10.4 and 10.5 respectively, much of the same trend may be seen: low values of CHT and high values of IDN result in high fitness values.

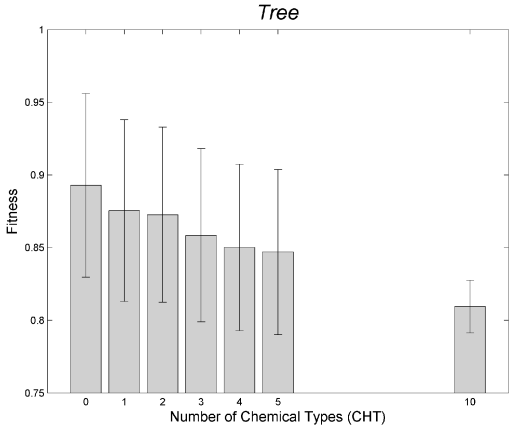


Figure 10.4: Fitness achieved when developing the tree with different numbers of chemicals (CHT)

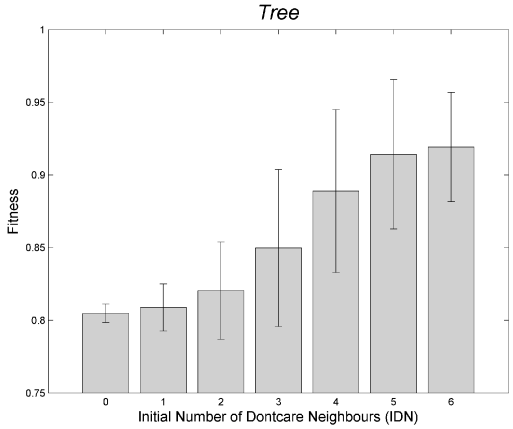
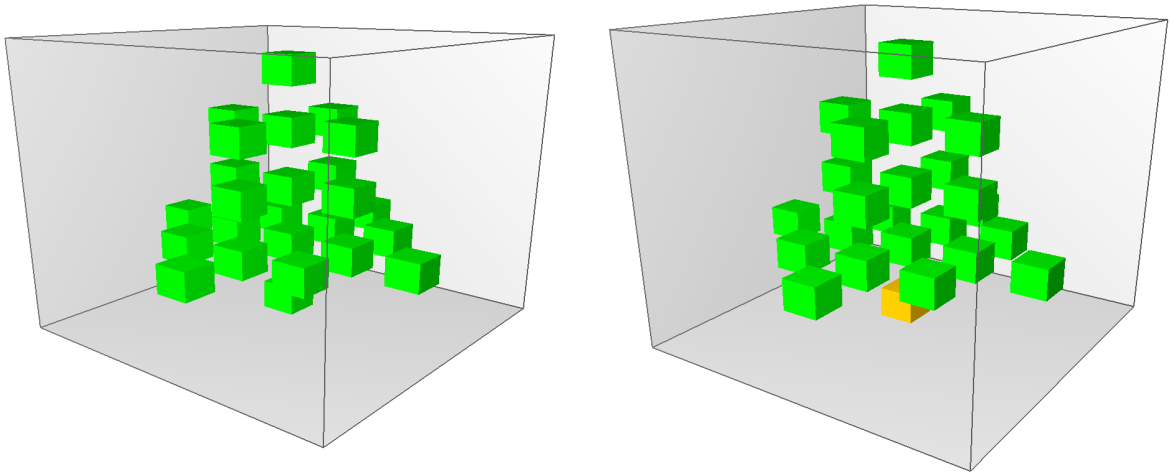


Figure 10.5: Fitness achieved when developing the tree with different initial number of don't care neighbours (IDN)

Compared to the cube, the standard deviation are here more varied. For CHT the standard deviations are almost equally large for zero to five and smaller for ten, while for IDN they increases from zero to tree, roughly equal from tree to five and slightly decreases from five to six. The small standard deviations for high values of CHT, respectively small value of IDN, may be because these values makes the search so difficult that not even by chance is the system able to evolve an organism with any higher fitness.

The four best evolved phenotypes were all identical: their shape had developed perfectly, the only error was the colour of the foot of the tree. The foot should have differentiated into an orange cell type, but instead it is green like the rest of the tree. See figure 10.6 for a comparison of one of the best evolved phenotypes with the target. An explanation for this error may be based on the facts that the dominant colour of the phenotype is green and the fitness function values correct shape higher than correct colour: the genetic programming algorithm may have been so focused on evolving genes responsible for the correct shape and for

differentiation to green that the genes for differentiation to orange simply have been “evolved away”.



Best evolved phenotype – fitness 0,996 **Target phenotype**

Figure 10.6: The only thing separating the best evolved phenotype (left) from the target (right) is the colour of the foot of the tree

10.3.3 Christmas Tree

As in the tree experiments, none of the experiments using the christmas tree as target phenotype was able to achieve perfect fitness. This is not surprising as these two phenotypes have exactly the same shape and the christmas tree requires more differentiation than the tree. Indications that differentiation is difficult to achieve was found in chapter 9.4.3. The two graphs plotting the fitness data with respect to chemical types and neighbours – see figure 10.7 and 10.8 respectively, display the same trends as was seen in both the cube and the tree experiments: low values of CHT and high values of IDN both result in high fitness values. The highest fitness achieved for the christmas tree phenotype was 0,98. This value was achieved in only one run.

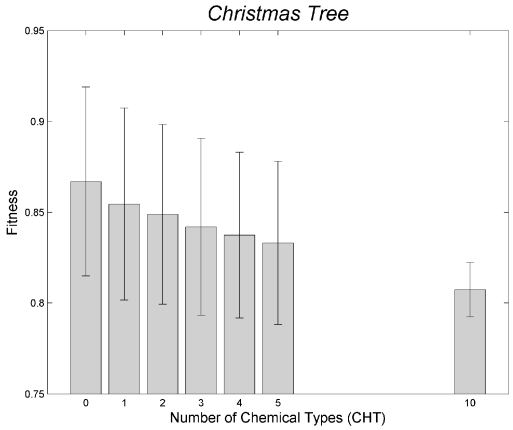


Figure 10.7: Fitness achieved when developing the christmas tree with different numbers of chemicals (CHT)

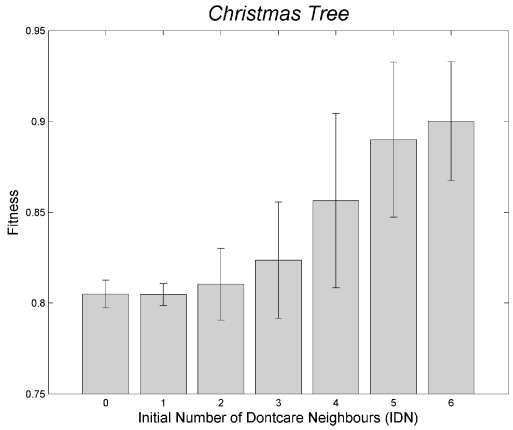
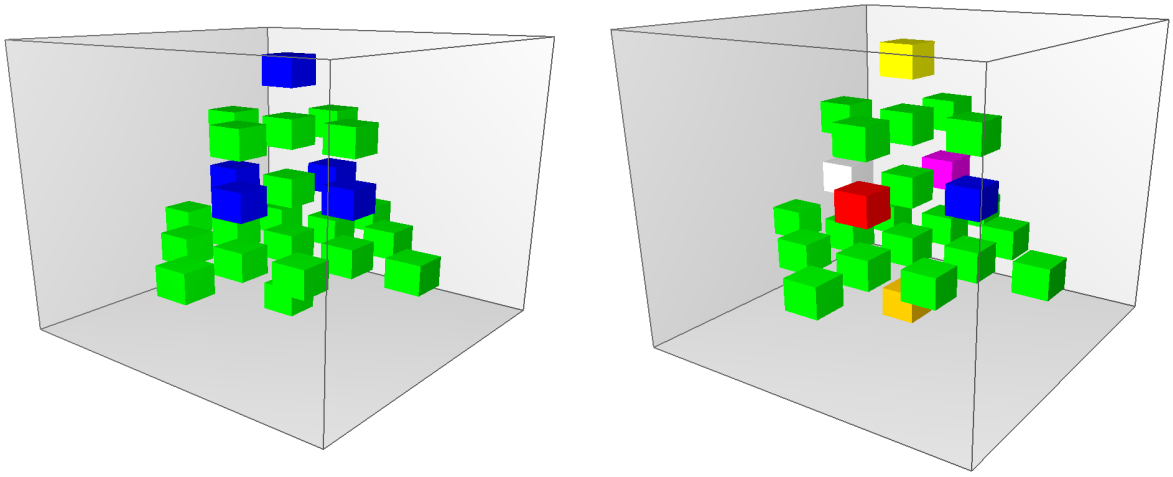


Figure 10.8: Fitness achieved when developing the christmas tree with different initial number of don't care neighbours (IDN)

No noteworthy differences was found in the standard deviations, compared to the tree. This may be an indication that developing shapes requiring more differentiation does not affect the systems ability to guide evolution. However, the support for this hypothesis is weak as the difference between the tree and the christmas tree is not large.

As with the tree, the shape of this phenotype was perfectly developed. The errors were that the cells which were supposed to differentiate to another colour than green, differentiated to the wrong colour: the foot of the tree became green instead of orange, and the star along with the "decoration" all became blue – see figure 10.9. It may seem a bit odd that the cells supposed to have another colour than green achieved this, but they got the wrong colour. However, the colour these cells got is blue, and as this is the colour of the zygote (the first cell in the organism) it may be that the organism first developed into shape and then differentiated the needed cells to green, just leaving the cells who did not need to be green as they were.



Best evolved phenotype – fitness 0,98 **Target phenotype**

Figure 10.9: Some of the cells in the best evolved phenotype (left) was unable to differentiate to the right type of cell, as specified in the target (right)

10.3.4 Sphere

Using the sphere as a target seems to have been a simple task for the system: perfect fitness was achieved in 856 out of the 980 experiment runs performed. Looking at the two graphs plotting the fitness achieved for various values of CHT and IDN – see figure 10.10 and 10.11 respectively, a light tendency of increased fitness with increasing IDN can be seen. For the chemicals, it is difficult to see any tendency at all. The only noticeable fact is that a CHT value of ten achieves much lower fitness than any of the other values tested. Looking at the standard deviations, however, it is seen that there are differences, also amongst the different CHT values: the standard deviation increases with increasing CHT. Just the opposite occurs with the standard deviations for the various IDN: they decreases with increasing IDN.

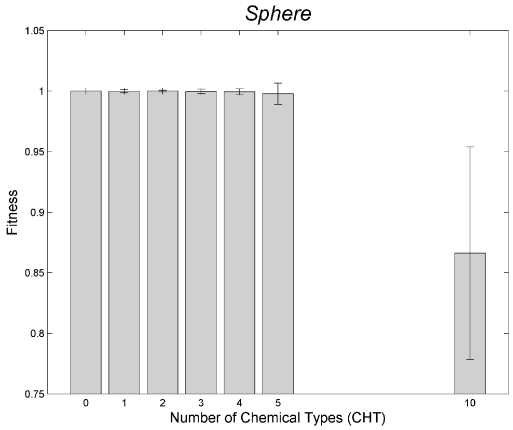


Figure 10.10: Fitness achieved when developing the sphere with different numbers of chemicals (CHT)

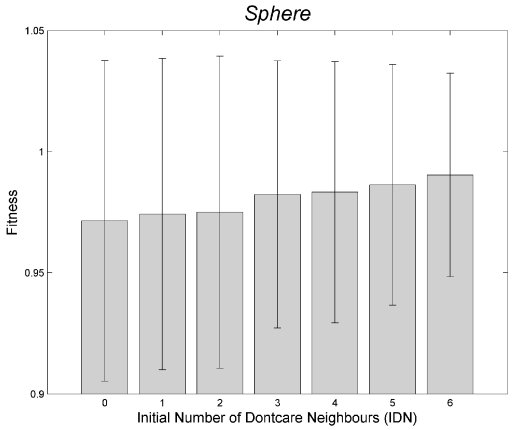


Figure 10.11: Fitness achieved when developing the sphere with different initial number of don't care neighbours (IDN)

The small differences with respect to the average fitnesses achieved and the relatively large differences between the different standard deviations, indicates that even though the achieved fitness is, on average, fairly equal regardless of CHT and IDN, achieving high fitness with high CHT or low IDN is much more by chance than with lower values of CHT, respectively higher values of IDN.

10.3.5 Divided Sphere

Even when introducing a different coloured plane inside the sphere, dividing it into two halves, the task seems to be a fairly simple one: 433 out of the 980 runs performed achieved perfect fitness. Compared to the experiments using the sphere as the target, the two graphs plotting the fitness achieved for various values of CHT and IDN for this phenotype – see figure 10.12 and 10.13 respectively, display a more marked tendency of the effect lowering the CHT value and increasing the IDN has on fitness. What is seen here is the same relation as was also seen in the results for the other phenotypes.

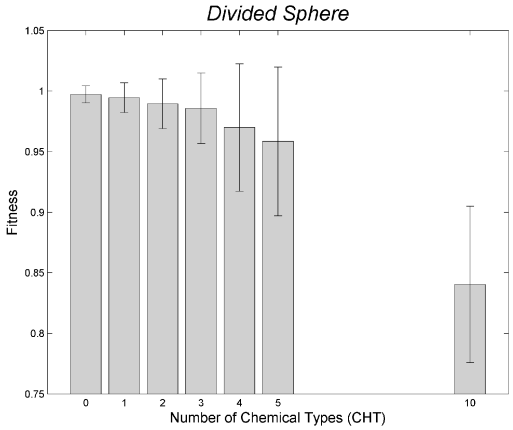


Figure 10.12: Fitness achieved when developing the divided sphere with different numbers of chemicals (CHT)

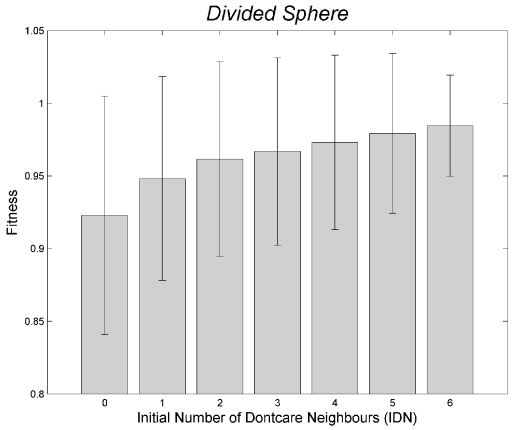


Figure 10.13: Fitness achieved when developing the divided sphere with different initial number of don't care neighbours (IDN)

As with the sphere, also here the standard deviations increases with increasing CHT, respectively decreasing IDN. This strengthens the hypothesis from chapter 10.3.3, that introducing more differentiation does not affect the systems ability to guide evolution.

10.3.6 Development

To gain a better understanding of how the development process actually goes about developing the zygote (the first cell) into the full grown organism, some of the DNAs evolved in this experiment series were chosen to be developed in a step-by-step fashion. A snapshot was taken at each discrete time step. The DNAs chosen, were the ones whose corresponding organism (phenotype) had the highest fitness. One DNA was chosen for each different target phenotype. When two or more shared the same fitness, one of them was chosen at random. The results can be seen in figure 10.14. Since the development process induced no change in the growing organism after six development steps, the steps seven to eleven are left out.

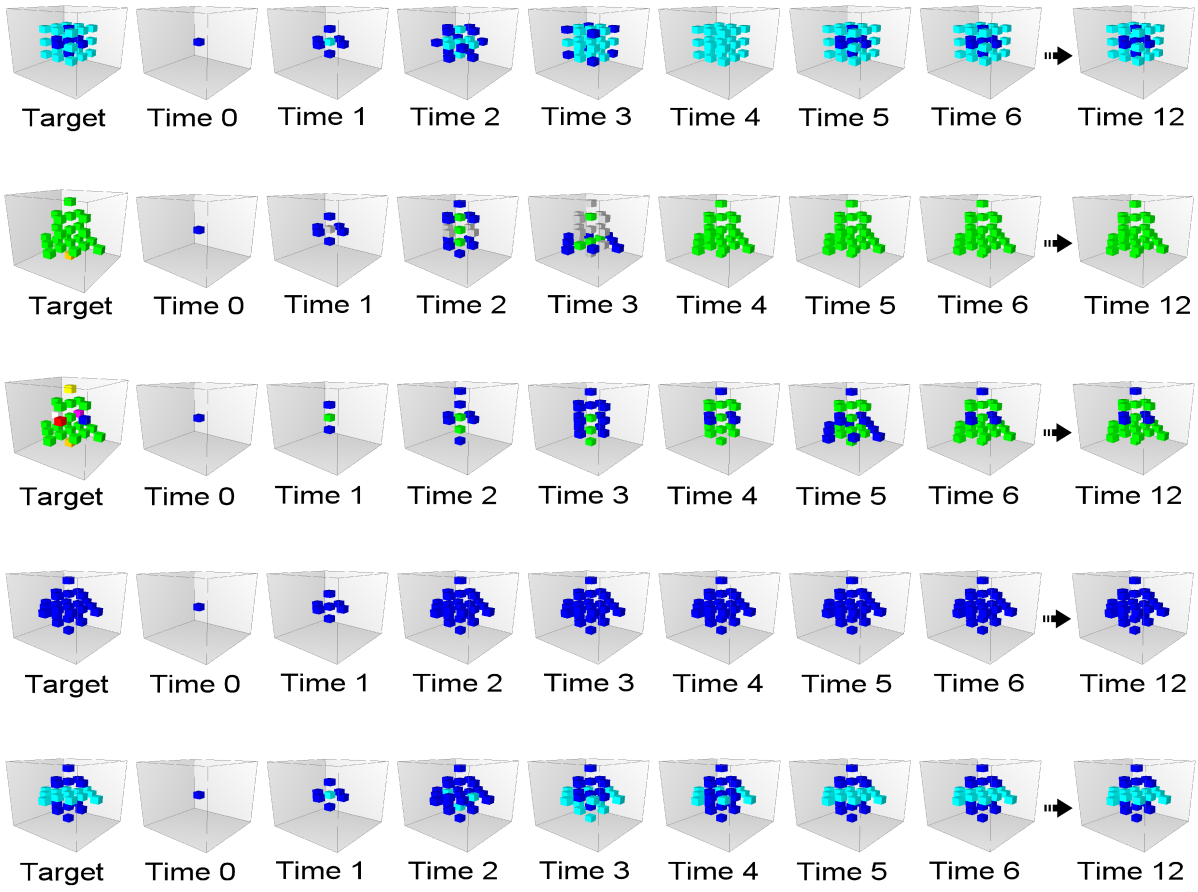


Figure 10.14: The development of the best evolved phenotypes, from zygote to full grown organism

What is common for all these developing organisms is that they all stop developing at a much earlier time than the expected one. Also, it seems that the more complex phenotypes requires more development time than the less complex ones. The longest development time (in the chosen ones) is six ticks, which was used by the DNA for the christmas tree phenotype. The shortest development time was used by the sphere, which only needed two ticks to grow into perfect shape. Why the organisms reach their final shape and colours at an early stage in development and do not develop any more after that, is not known. It may happen because of the limited number of ticks each protein lives: when there are no more proteins alive in the organism, no further development will occur. Another possibility is that, as the organisms developed fully earlier than anticipated, after 12 ticks only the organisms that stopped developing at an early stage achieve high fitness. Those who did not stop to develop would at the 12^{th} tick, when the fitness is measured, have developed away from their maximum closeness to the target, and thus receive low fitness. If this hypothesis is correct, this puts a larger burden on evolution to come up with a good DNA, as only those who stops their own development will achieve high fitness. This finding is an indication that the chosen formula for estimating the required development time may need refinement. However, no experiments have been performed to investigate what the optimal number of development steps actually is. Also, the achievements of ArtDev3D so far is satisfactory, so no changes to the formula for calculating development time i.e. the number of ticks, is performed.

10.4 Conclusion

This experiment series has shown that ArtDev3D is capable of developing a number of different phenotypes. The investigation of the effect different values for CHT and IDN have on the systems ability to evolve good phenotypes indicated that increasing CHT, respectively decreasing IDN, resulted in a decrease in the achieved fitness. Investigation of the randomness of the achieved fitness (achieving high fitness by "luck"), as measured by the standard deviations, indicated that increasing CHT, respectively decreasing IDN, results in more randomness for the simpler shapes, while opposite, less randomness, for the more complex shapes. This may be because when evolving simpler shapes, it is easier for to guide the evolution correctly, resulting in stable and high results. When evolving more complex shapes the guiding of evolution is more difficult, resulting in a need for luck to evolve highly fit individuals. However, it is difficult to draw any conclusions without further experiments. Additionally, some support for the hypothesis that the amount of differentiation does not influence the systems ability to guide evolution was found. Further investigations regarding CHT and IDN's effect on the achieved fitness will be conducted in the following chapter.

It was also found that the time required for successful development may have been set too high and that complex target phenotypes may require longer development time than the less complex ones. The long development time revealed, however, an interesting aspect of the system: it is capable of producing organism which remains stable i.e stops developing, after they reach maturity.

11 Experiment 2

The initial testing of ArtDev3D – see chapter 9, showed that the system was able to perform the three required features growth, differentiation, and morphogenesis. Further experiments, see chapter 10, established that the system was capable of developing a number of different phenotypes and investigated the effect of changing two of the systems parameters. The results from these experiments are promising, and now it was time for a look into the finer details of how the system works.

11.1 Description

Two development parameters and two phenotype parameters are thought to play a key role in affecting the system's ability to achieve high fitness. The two development parameters in question are parts of a proteins precondition: the initial number of don't care neighbours (IDN) and the number of different chemical types (CHT) to use. The two phenotype parameters in question are the target phenotype's shape and colours (as given by the cell type). What was tested for in this experiment series was both the degree of symmetry (DoS) and the number of colours (NoC) used for the target. The four parameters have been tested using an array of different values for each one of them.

As indications were found in the previous experiment series that low values for IDN and high values for CHT resulted in low fitness, some of the least promising values for these two parameters were changed or left out in this experiment series.

11.2 Setup

To investigate these four parameters, 300 single experiments were performed. These include experiments for all possible combinations of zero, two, four, five and six IDN, zero, one, two, four and eight CHT, and 12 different target phenotypes (four different shapes – zero to three DoS, each using two, four or six NoC)

All experiments were repeated 20 times.

11.2.1 Experiment arrangement

This experiment series can be divided into four groups based on the degree of symmetry (DoS) of the phenotype:

- 0 DoS, shape: Blob
- 1 DoS, shape: Airplane
- 2 DoS, shape: Tree
- 3 DoS, shape: Diamond

Each of these four can then be further divided into three subgroups, based on the NoC value: two, four and six colours.

Each of the above groups can again be divided into five subgroups, based on the IDN value, and each of these can be divided into five single experiments, based on the CHT value.

This arrangement of the experiments ensures that only one parameter is changed from one experiment to the next (see table 11.1).

Experiment no.	***Degree of Symmetry (DoS)***	***Number of Colours (NoC)***	***Initial Number of Don't Care Neighbours (IDN)***	***Number of Chemical Types (CHT)***
1	0	2	0	0
2				1
3				2
4				4
5				8
6 – 10			2	0, 1, 2, 4, 8
11 – 15			4	0, 1, 2, 4, 8
16 – 20			5	0, 1, 2, 4, 8
21 – 25			6	0, 1, 2, 4, 8
26 – 50		4	0, 2, 4, 5, 6	0, 1, 2, 4, 8
51 – 75		6	0, 2, 4, 5, 6	0, 1, 2, 4, 8
76 – 150	1	2, 4, 6	0, 2, 4, 5, 6	0, 1, 2, 4, 8
151 – 225	2	2, 4, 6	0, 2, 4, 5, 6	0, 1, 2, 4, 8
226 – 300	3	2, 4, 6	0, 2, 4, 5, 6	0, 1, 2, 4, 8

Table 11.1: Experiment series setup

11.2.2 Parameters

The same parameters were used in all experiments except for the four parameters to be investigated: initial number of don't care neighbours (IDN), number of chemical types (CHT), number of colours (NoC), and degree of symmetry (DoS).

Genetic Algorithm

The following are the parameters used by the genetic algorithm during this experiment series.

General settings

Population size =	1000
Crossover rate =	0.9
Mutation rate =	0.1
Termination criterion =	Generations=500 OR fitness=1.0
Fitness function =	*CellByCellComparisonFitness*
Selection method =	*TournamentSelection* • group size = 4

Genotype specification

Promotor =	6 bits
Initial no genes =	5
Proteins used =	*GAdivideCellProtein* *GAtranscribeGeneProtein* • length of subpromotor = 3 bits *GAchangeTypeOfCell* • number of different cell types = {2, 4, 6}
Proteins	
Max time to live =	5 ticks
Precondition =	Chemicals – number of different types = {0, 1, 2, 4, 8} Neighbours – initial number of don't cares = {0, 2, 4, 5, 6}

Target phenotypes

Four different shapes, each with a specific degree of symmetry, are used as target phenotypes:

- Blob – 0 DoS
- Airplane – 1 DoS
- Tree – 2 DoS
- Diamond – 3 DoS

Each of the shapes has either two, four or six colours, making a total of 12 phenotypes

All the phenotypes have the same dimensions: seven-by-seven-by-seven units. The size of the phenotypes were increased, with respect to the previous experiment series, to allow the use of six cell types while maintaining the required symmetry.

Development

All the experiments use the same development time of 17 ticks. The formula used to calculate the number of ticks is the same as in the previous experiment series - see chapter 10.1.1.

11.2.3 Target phenotypes

Following is a graphical presentation of the 12 different phenotypes used as targets in these experiments (exploded view).

Zero Degrees of Symmetry (Blob)

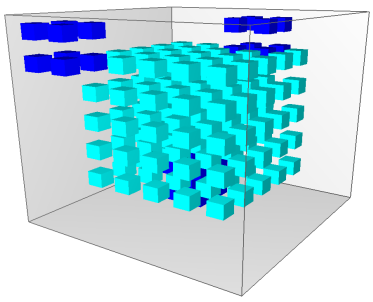


Figure 11.1: Blob with two colours

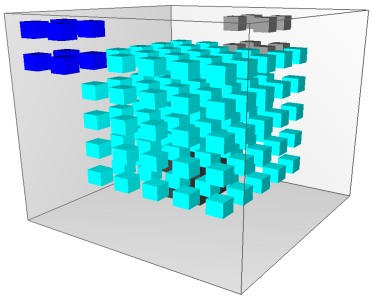


Figure 11.2: Blob with four colours

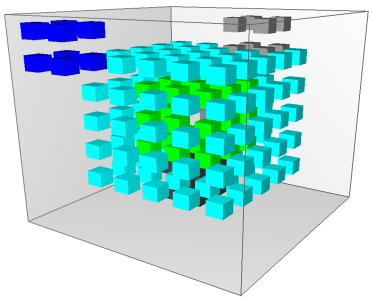


Figure 11.3: Blob with six colours

One Degree of Symmetry (Airplane)

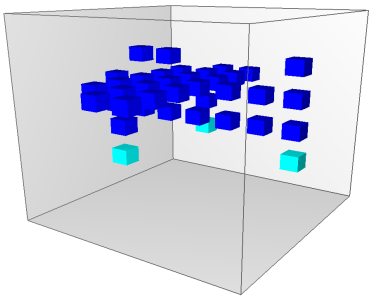


Figure 11.4: Airplane with two colours

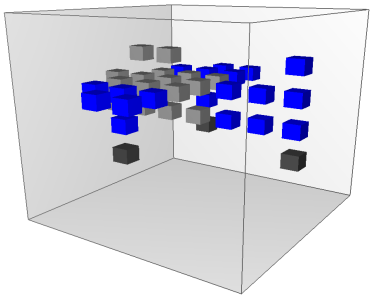


Figure 11.5: Airplane with four colours

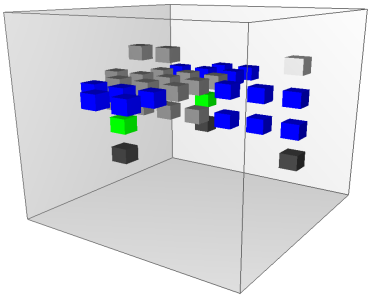


Figure 11.6: Airplane with six colours

Two Degrees of Symmetry (Tree)

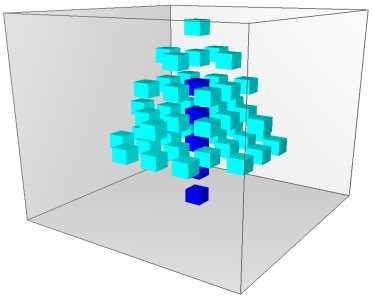


Figure 11.7: Tree with two colours

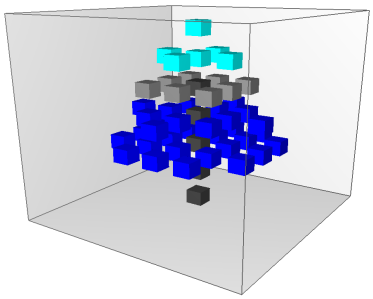


Figure 11.8: Tree with four colours

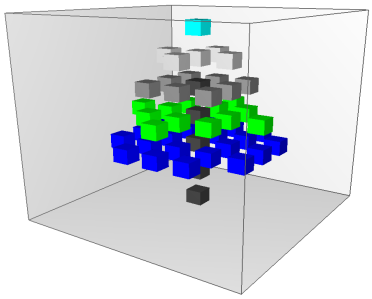


Figure 11.9: Tree with six colours

Three Degrees of Symmetry (Diamond)

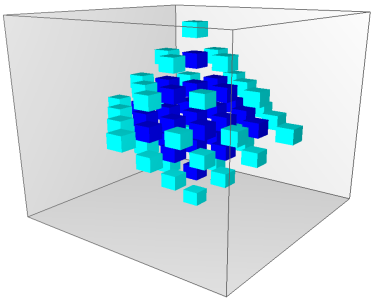


Figure 11.10: Diamond with two colours

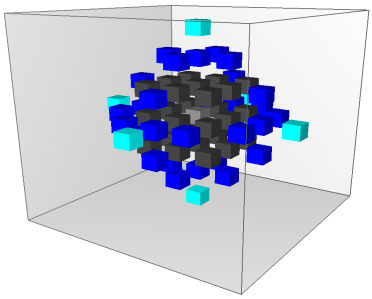


Figure 11.11: Diamond with four colours

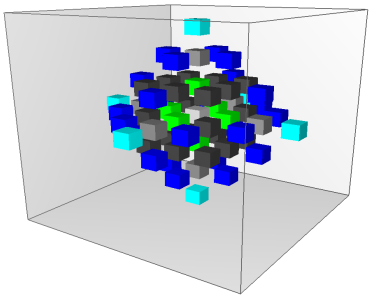


Figure 11.12: Diamond with six colours

11.3 Results

A significant amount of data is generated with so many development experiments. It was, therefore, important to select which data should be stored for each experiment, providing sufficient information to enable the analysis sought to be undertaken. The following data was used for this analysis: the final fitness of the best evolved individual in each repetition of each single experiment, along with the DNA used to construct it, and the best fitness score for each generation.

The final fitness was used to plot graphs to visualize the effect a parameter has on the achieved fitness.

The DNA was used to determine the average number of don't care neighbours in the best individuals. This was then used for comparison with the initial number of neighbours to see if any patterns appear.

The best fitness for each generation was used to analyse the genetic algorithm. Using this data it can be investigated how the evolution proceeds, making it possible too see, for example, whether it is likely that increasing the number of generations will result in higher fitness.

The parameters were analysed one by one. Because IDN and CHT are parameters that can be changed at will according to the wanted phenotype (which is specified by shape and colours), they was analysed first. After these two, DoS and finally NoC were analysed.

11.3.1 The Initial Number of Don't Care Neighbours

The first parameter to be investigated was the initial number of don't care neighbours (IDN) – see chapter 11.3.2. The allowed values for IDN ranges from zero to six. Experiments were run for IDN values of zero, two, four, five, and six. IDN values of one and three were left out of this series to save computation time as the results from the previous experiment series – see chapter 10, indicated that little information could be gained by including also these two values.

General effect

To find the general effect on fitness when changing the IDN, the data was plotted as a bar graph with standard deviations on top. The data was obtained using the highest fitness achieved in every run, and for every experiment. This was then organized in such a way that the various values of IDN could be plotted with their corresponding average and standard deviation values. The resulting graph – see figure 11.13, shows that the average of the best fitness is higher for high values of IDN than for low values, and that the average fitness increases with increasing IDN. The increase seems to be almost linear. The differences in fitness are relatively small, the lowest is 0,83 and the highest 0,91. With respect to this, the standard deviations are large, which makes it difficult to draw any valid statistical conclusions.

The graph gives the impression that there exists a positive correlation between IDN and fitness. The fact that the increase is almost linear indicates that the correlation is steady and robust. To test this impression for statistical significance, a one-way ANOVA test was employed to compare group means. Two assumptions need to be fulfilled to perform such a test: the groups must both be normally distributed and have equal variance. The ANOVA test was chosen because it is robust to modest violations of these two assumptions. This robustness is important because the experiments were only repeated 20 times each, a number that is generally too small to fulfil these two assumptions.

The results obtained from testing the data at a 95 % confidence interval strengthens the hypothesis that there exists a positive correlation between fitness and IDN. It shows that there are indeed statistically significant differences among the group means. However, this does not mean that a positive correlation has been proven, only that changing IDN has an effect on fitness. The graph is used as a visual aid to get an idea of the direction of this effect.

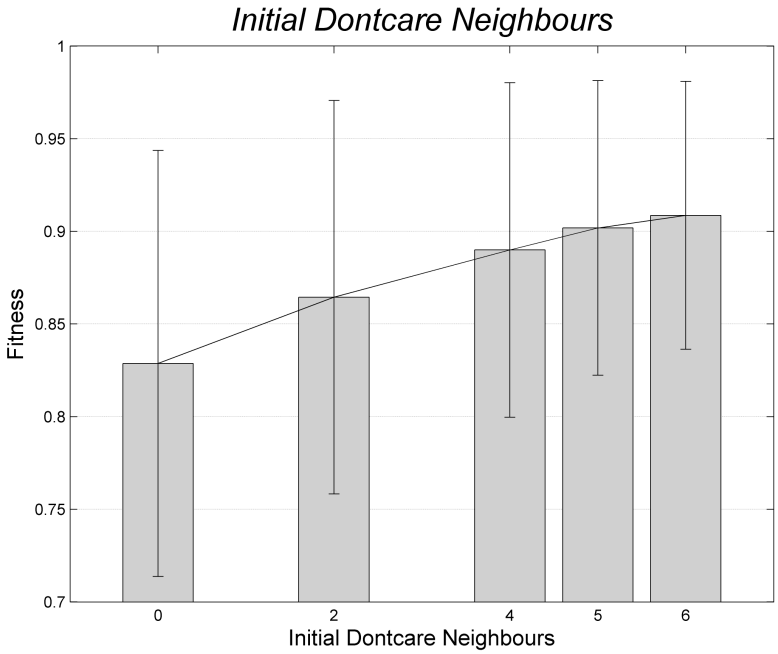


Figure 11.13: Fitness achieved using various values for IDN

According to the findings so far, the highest IDN should be chosen to achieve highest possible fitness. Is it really that simple? Not necessarily. Although the graph seems to imply this correlation, and the ANOVA test confirms statistically significant differences between the group means, the standard deviations are relatively big and the correlations with the other three parameters are not known. This means that there might be other, maybe more complex, correlations between IDN and fitness, given various values for the other parameters.

To get a clearer picture of the relationship between IDN and fitness, three more graphs were created, each showing this relationship given each of the other three parameters. To make the data easier to analyse visually, the graphs were plotted as dots with connecting lines, and the standard deviations were left out. As CHT is the most interesting of the three parameters, IDN is first analysed against this one, followed by DoS and NoC.

Initial Number of Don't Care Neighbours vs. Number of Chemical Types

The graph plotting IDN vs. CHT – see figure 11.14, indicates that when taking the number of chemical types into account, the relationship between IDN and fitness becomes more complex.

The curves for low values of CHT resembles a cut-out in the upper left corner of a bell-shaped curve. There is an increase in fitness for IDN values up to five, where it has its top, and then a decrease from five to six. The curve for one chemical type shows a small irregularity from this description at two IDN. For higher values of CHT, the curve resembles a sigmoid function (CHT=4), and an exponential function (CHT=8).

For almost all values of CHT, there is a relatively rapid increase in fitness for some value of IDN. For low values of CHT, this rapid fitness increase appears for low values of IDN, but there seems to be an increase in the IDN required for this rapid fitness increase with increasing CHT.

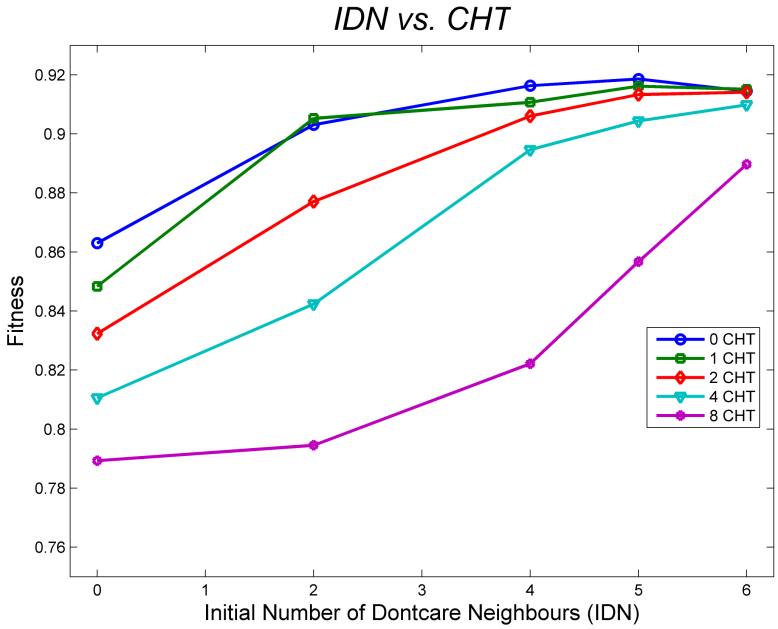


Figure 11.14: Fitness achieved with various values for inital number of don't care neighbours (IDN) and number of chemical types (CHT)

Difficult to see in the above graph alone, but more evident when comparing an array of graphs where IDN vs. CHT is plotted for each of the values for NoC and DoS, is that the graph seems to resemble a bell-shaped curve which shifts right and left with increasing and decreasing values of CHT – as illustrated in figure 11.15. When looking at the fitness achieved with respect to IDN for a given combination of CHT, DoS and NoC, only a part of this hypothesised bell-shaped curve comes into view – see figure 11.16. If this really is the case, the way to achieve the highest possible fitness will be to adjust the value for CHT so the top of the bell-shaped curve comes into view (i.e. the top lies within zero-six IDN), and then choose the value for IDN where the top is.

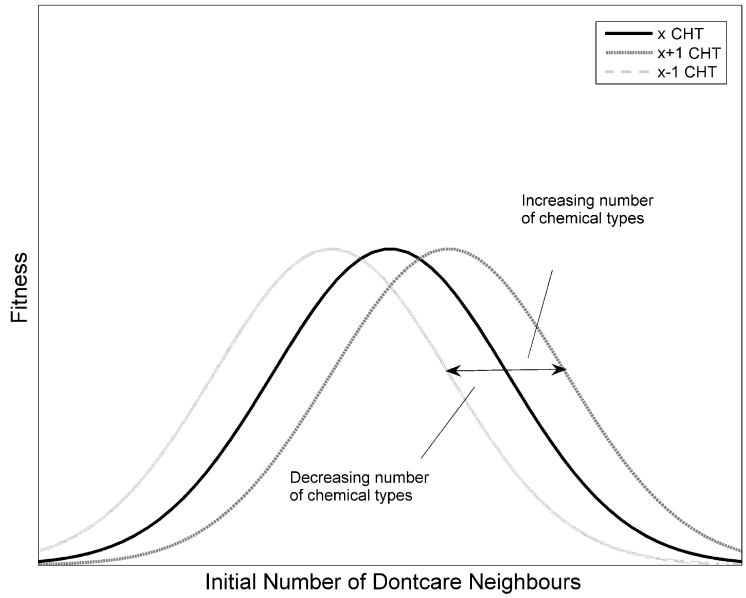


Figure 11.15: Shifting of the bell-shaped curve according to changes in the number of chemical types used

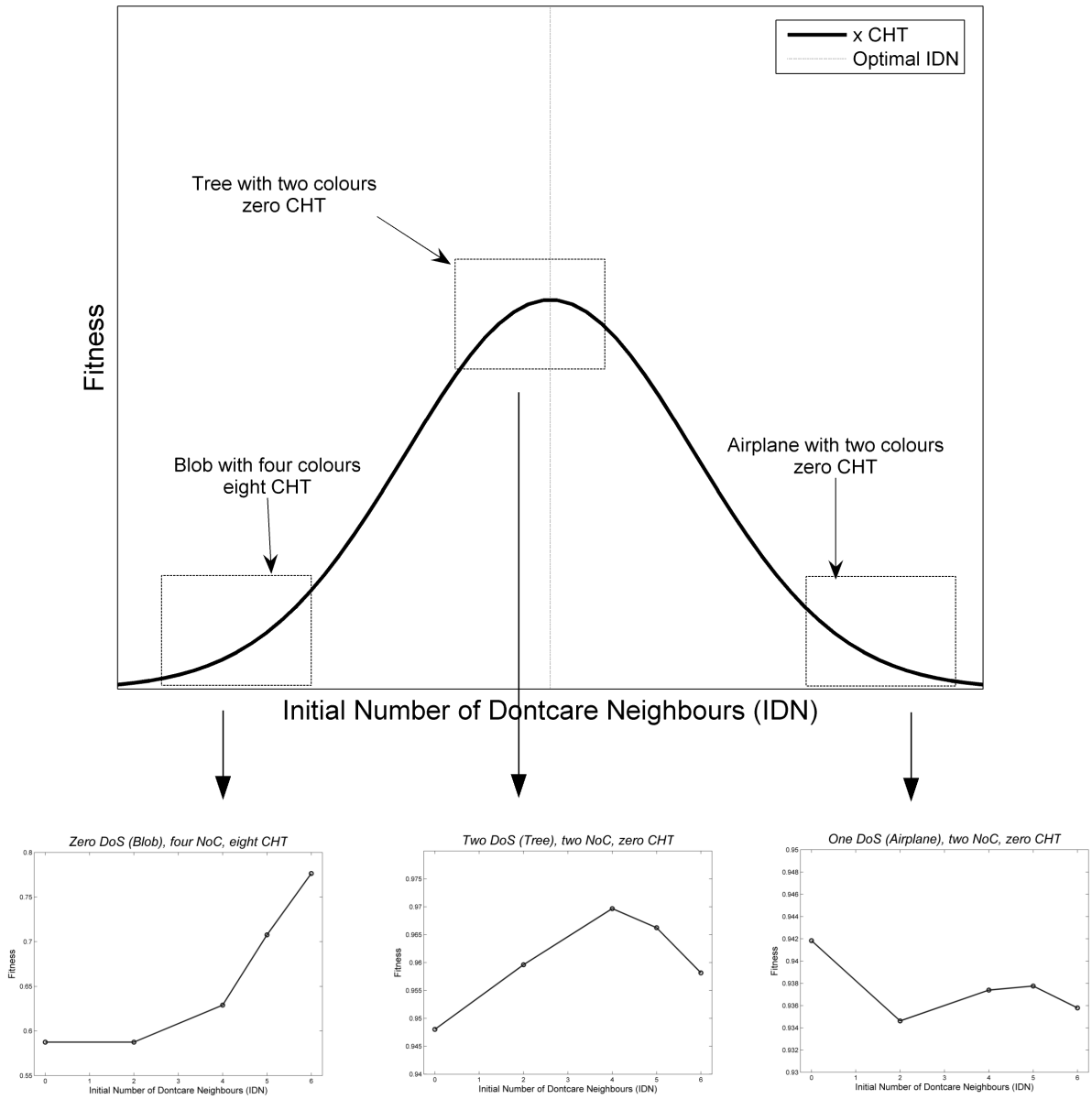


Figure 11.16: Examples of graphs supporting the hypothesis of a bell-shaped curve

The investigation of graphs plotting IDN vs. CHT for each value of NoC and DoS, showed no noteworthy deviations from the graph plotted using the averages.

Initial vs. Final Number of Don't Care Neighbours

So far it was shown, for the experiments performed, that when disregarding the values for CHT, NoC and DoS, fitness increases with IDN. This implies that using six IDN would be the best choice when these three parameters are unknown. However, this does not mean that taking the neighbourhood into account (i.e. when the number of don't cares is less than six) is a bad choice. There are two important reasons for this. First, the value of CHT can be set manually,

meaning this parameter need not be unknown. And, as has been shown, when CHTis taken into account the relationship between IDN and fitness becomes more complex, often leaving six IDN with less fitness than five IDN. Second, as IDN denotes the *initial* number, the number of actual don't care neighbours may well change during evolution. It might be the case that the optimal number of don't care neighbours lies below six, and that it is just easier for the genetic algorithm to evolve the correct neighbourhood when starting with many don't cares.

To look into the possibility that the optimal number of Final Don't care Neighbours (FDN) is different from the optimum for IDN, the two variables were plotted as arrows, one for each different IDN. The starting point for the arrow is the IDN value, and the ending point is the average FDN for this IDN. The boxes represent the range (minimum and maximum values) of different FDN values evolved. This way to plot the data makes it easy to see both how big the change in number of don't care neighbours is, the direction of the change, and how much the FDN varies. The latter is important because it gives an impression of how much the evolutionary algorithm is able to vary the number of don't care neighbours given an IDN. The FDN is calculated as the average number of don't care neighbours in each of the best individuals evolved.

As can be seen in figure 11.17, the number of don't care neighbours seems to be moving towards an overall optimum, which seems to lie just above four don't cares. For IDN values of four or less, the FDN is greater than IDN, and for IDN above four, smaller. Also, the further away from the assumed overall optimum the IDN is, the greater the change. The range of FDN is larger for low IDN's than for high ones, possibly suggesting that low values of IDN make it easier to evolve the best number of don't care neighbours. However, when looking closer at 0 IDN it becomes evident that the average is low, meaning only a few individuals evolves to have many don't care neighbours. Also, it was established that low values of IDN generally result in lower fitness, meaning high values of IDN may have a smaller range because increasing their range implies evolving less fit individuals. Taken together with the finding that, on average, higher IDN results in higher fitness – see chapter 11.3.1, the hypothesis that low IDN makes the evolution easier doesn't seem plausible.

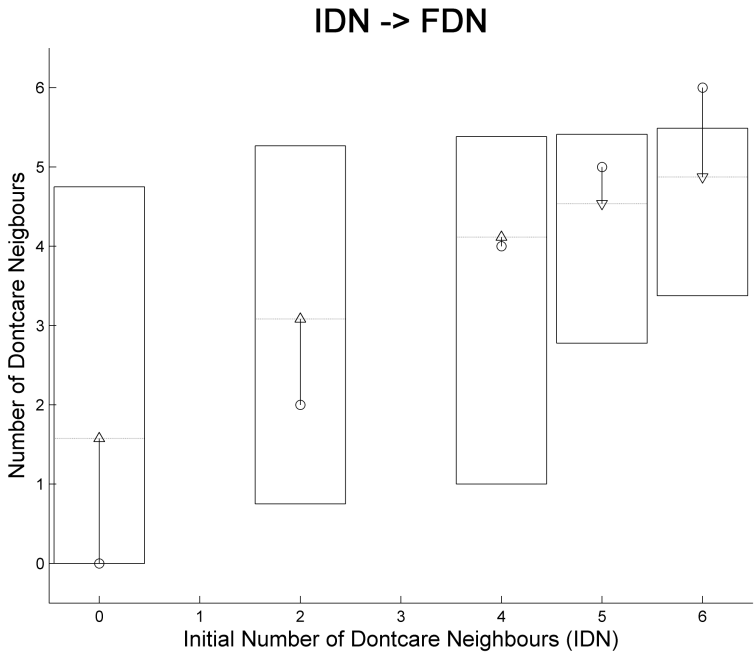


Figure 11.17: Changes in the number of don't care neighbours from before (IDN) to after (FDN) evolution

So far it was found that the number of don't cares seems to move towards a global optimum. However, the plot uses averages and min-max values, making it difficult to see how the FDN is distributed. To get a better understanding of what is going on, it is useful to look at a visual representation of this distribution. When plotting the data as a histogram – see figure 11.18, it is easy to see that an FDN value of four or five (i.e. between 3,5 and 5,5) is the most frequent one with 30,75 % and 34,56 % of the occurrences respectively. They are both more than twice as frequent as the third-most frequent one (three FDN, occurring 12,97 %).

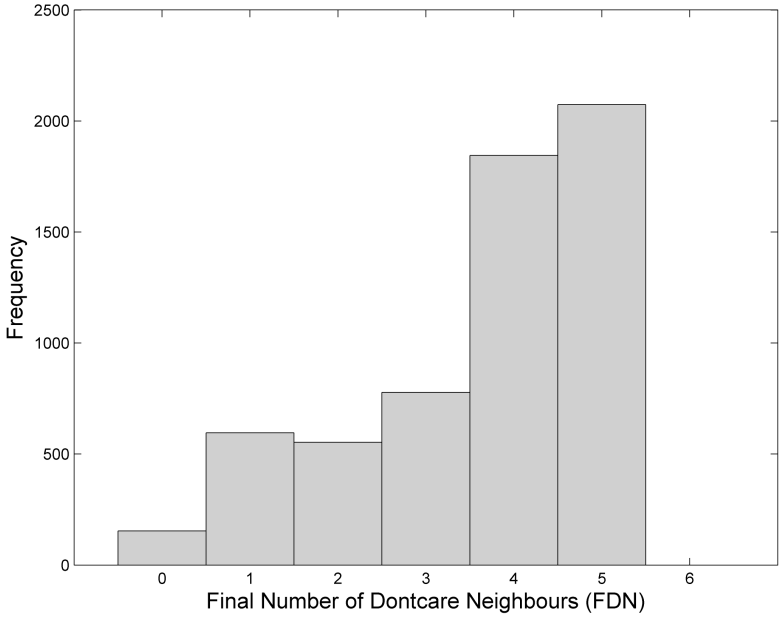


Figure 11.18: Histogram showing the frequency distribution of final don't care neighbours (FDN)

Plotting histograms of FDN distribution for specific DoS, NoC and CHT does not reveal any large deviation from this "collective" histogram. Some have the highest frequency for four FDN, while others have five FDN as the most frequent – see figures 11.19 and 11.20. What seems to be the main theme is that four and five FDN are the two most frequent ones.

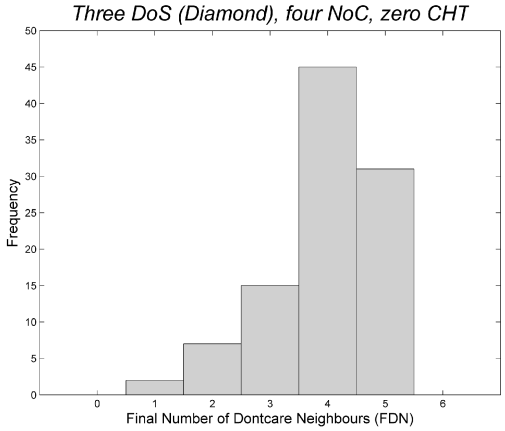


Figure 11.19: Three DoS (Diamond) with four colours has four FDN as the most frequent one when not using any chemicals

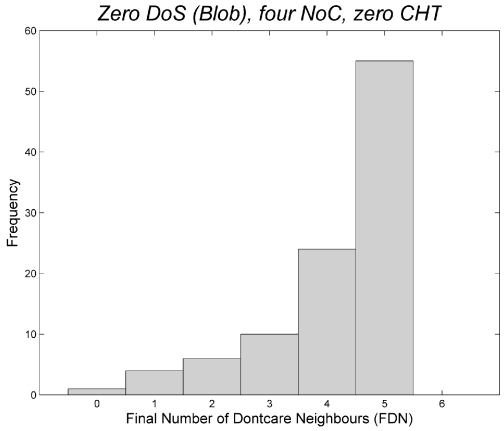


Figure 11.20: Zero DoS (Blob) with four colours has five FDN as the most frequent when not using any chemicals

However, some deviations do exists. Some plots have a bimodal distribution, others have a more even distribution, and for some the rise in frequency is almost linear with increasing FDN – see figures 11.21, 11.22 and 11.23 respectively. One explanation for the first deviation might be the values chosen for IDN in the experiments. No experiments where performed for one and three IDN. Low values of IDN generally results in low fitness, and it may be the case that because low fitness makes it difficult to guide the evolution in the correct direction, the genetic algorithm has a problem "evolving away" from two don't care neighbours. Also, zero and four IDN were found to move towards higher FDN. The even distribution was found in experiments which are typically difficult to evolve. This difficulty may mean that the search was more or less random, making the value of FDN unimportant; the fitness was bad even for high numbers of don't cares. The linear rise in frequency is harder to explain. It might be a sign of "good" evolution of the neighbourhood, meaning the evolutionary algorithm is able to freely evolve a good number of don't cares given the IDN, and that for the number of don't cares there is no threshold where the fitness suddenly gets much higher (like the ones seen in figure 11.18).

The two latter were found only in a few experiments, and they don't deviate that much from the "collective" histogram, meaning they might well be no more than random fluctuations.

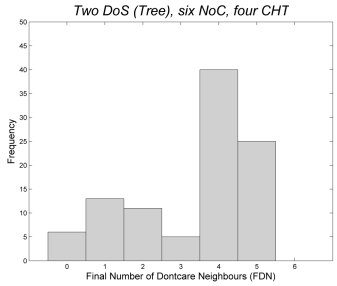


Figure 11.21: Bimodal distribution of FDN found for two DoS (Tree) with six colours, using four chemical types

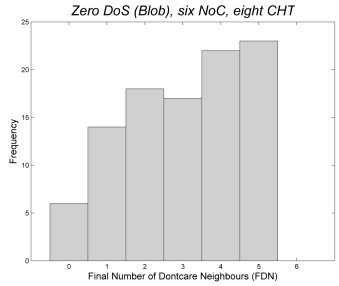


Figure 11.22: A, to some extent, even distribution of FDN was found for zero DoS (Blob) with six colours, using eight chemical types

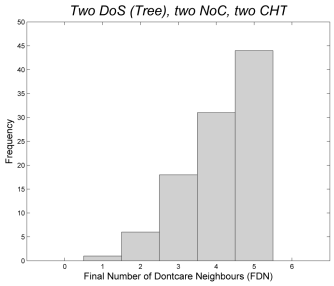


Figure 11.23: The distribution of FDN for two DoS (Tree) with two colours, using two chemical types, shows a close-to linear rise in frequency with increasing FDN

Initial Number of Don't Care Neighbours vs. Degrees of Symmetry

When plotting IDN vs. DoS – see figure 11.24, the relationship between IDN and fitness also becomes more complex, although not in the same way as with IDN vs. CHT. Here it seems like the value of DoS has an effect on how important IDN is for the fitness, rather than indicating what the optimal value for IDN is.

The graph shows an almost linear relationship between IDN and fitness for all values of DoS. The slope of the curve is pointing upward for all values, but the steepness of the slope is dependent on the value of DoS. For one to three DoS the slope becomes steeper with increasing DoS, and the lines cross each other at two IDN. The line for zero DoS is somewhat special. Its slope has about the same steepness as the slope for three DoS, but the fitness values are much lower.

All the lines are rising. This indicates that increasing IDN has a positive effect on fitness regardless of the DoS, and that six IDN is the optimal value. The latter must not be taken as an absolute fact, as it was found above that the value of CHT has a great influence on which value of IDN is the optimal one. The varying degree of steepness of the lines indicates how important the choice of IDN is for fitness, given the DoS. A steep slope indicates the choice of IDN has a big influence of the fitness, while a more moderate slope indicates a weaker influence. For this graph this means that for zero and three DoS, the choice of IDN will have a greater impact on the resulting fitness (max fitness gain: 0,13 and 0,10) than for two and one (max fitness gain: 0,06 and 0,03).

If disregarding the line for zero DoS, the graph also seems to indicate that increasing DoS increases the importance of the value of IDN. One reason for zero DoS to fall outside of this assumed pattern may be the shape chosen to represent zero DoS. This shape is essentially a cube, embedded with small cubes here and there to make it asymmetric. As this shape generally achieved low fitness in the experiments, it may be the case that the algorithm was never able to evolve a shape that was more than a cube. If this is so, then it would explain why the line for zero DoS resembles that of three DoS. After all, a cube is a shape with three DoS. This issue is given more attention in chapter 11.3.3.

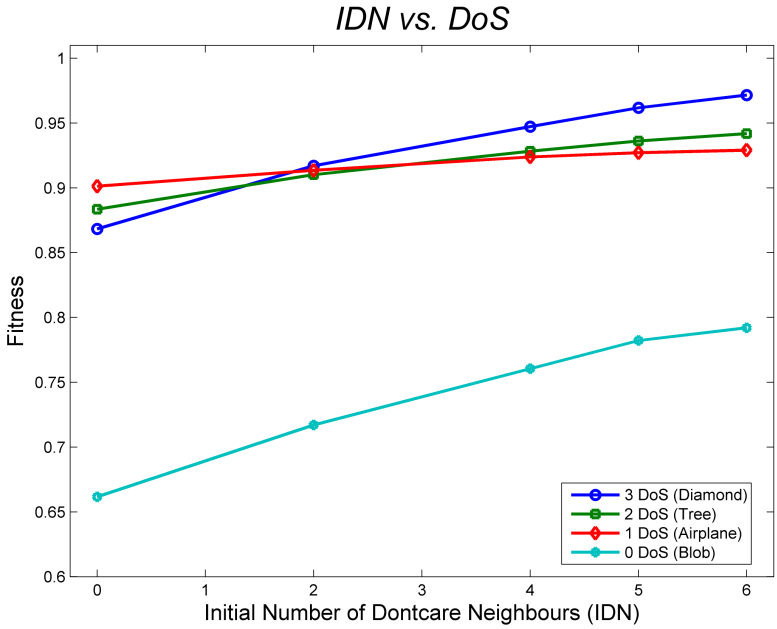


Figure 11.24: Fitness achieved with various values for initial numbers of don't care neighbours (IDN) and degrees of symmetry (DoS)

Plotting IDN vs. DoS for each value of CHT and NoC (not shown), does not reveal much new information. For the various values of NoC, the relationship between IDN and fitness is still linear, giving the same visual impression as the IDN vs. DoS plot. For CHT, the lines are not linear, but they resemble the lines found when analysing IDN vs. CHT. This is as expected. What is interesting, however, is the change in IDN value for the crossing point between the

lines for one, two and three DoS. With increasing CHT, the crossing-point moves to higher IDN, and the same happens for increasing NoC. To give a good answer to what this could indicate is difficult. What both increasing CHT and NoC have in common is the fact that they make the evolution process more complicated. It seems like the slope of the line for one DoS remains unchanged, while two DoS becomes somewhat steeper with increasing CHT or NoC, and three DoS even more steep. At the same time all lines shifts downward to lower fitness. Three DoS has a bigger decrease in fitness than two DoS, and one DoS decreases the least. Taken together, this may indicate that the value for IDN becomes more and more important not only for increasing DoS, but also for increasing CHT or NoC.

Initial Number of Don't Care Neighbours vs. Number of Colours

To complete the picture we have been getting so far as to how IDN influences the system, IDN was plotted against fitness for various NoC – see figure 11.25. The graph shows three almost parallel lines. All lines are rising, and they are all close to being straight lines, except for each having a slight left-hook. The line for two colours deviates to some extent from the shape of four and six colours.

The fact that the lines run almost in parallel indicates that the number of colours used doesn't affect the complexity of the relationship between IDN and fitness much.

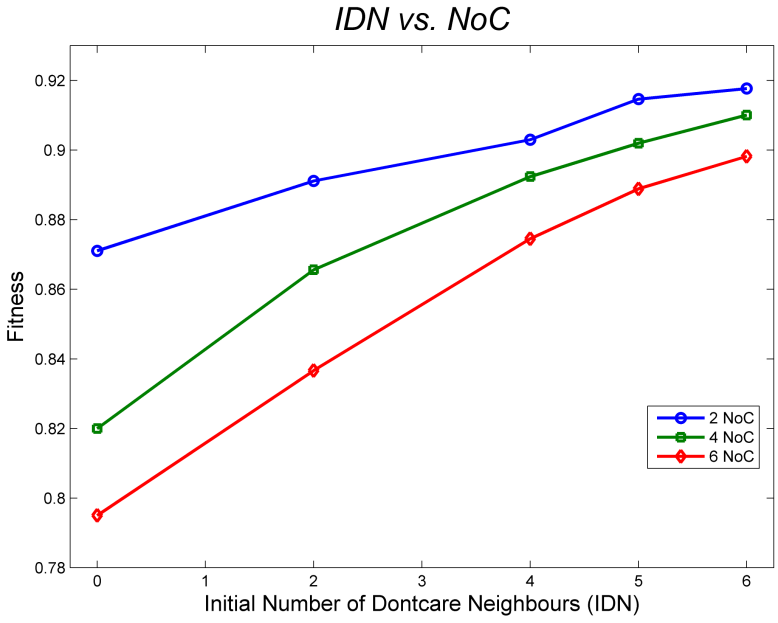


Figure 11.25: Fitness achieved with various values for initial number of don't care neighbours (IDN) and number of colours (NoC)

Plotting IDN vs. NoC for each value of CHT and DoS does, however, reveal that the picture is more complicated after all. Although the plots resemble the one using averages for most values

of CHT and DoS, there are two noticeable deviations, namely when plotted using four chemical types, and when using two degrees of symmetry.

The deviation found when using four chemical types is not that large – see figure 11.26, but it seems clear that different numbers of colours leads to different effects on the relationship between IDN and fitness. While the lines for the three values of NoC have an almost constant distance between them when plotted using averages, this is not the case for four chemical types. In this plot the curve for two colours is close-to linear with a slight left hook, while the curves for four and six colours resemble a sigmoid function. For IDN values of four, five and six, two and four colours overlap, achieving almost equally high fitness.

What may have happened is that evolution was simply unable to achieve high enough fitness for two colours to match the steep increase in fitness from two to four IDN which occurred for four and six colours. This is based on the observation that for zero to two IDN all three lines are close-to parallel, but from two to four IDN, four and six colours have a steep increase in fitness which two colours fails to follow. For four to six IDN, they are all parallel again. Why is evolution able to achieve this steep increase in fitness for four and six colours, but not for two colours? As two colours achieves highest fitness regardless of IDN, it may be that with four and six colours and IDN values of zero and two it is just too difficult to evolve something good, but suddenly, when the search becomes easier (IDN values equal to or greater than four) fitness increases rapidly. Fewer colours should be easier to evolve so this explanation seems plausible.

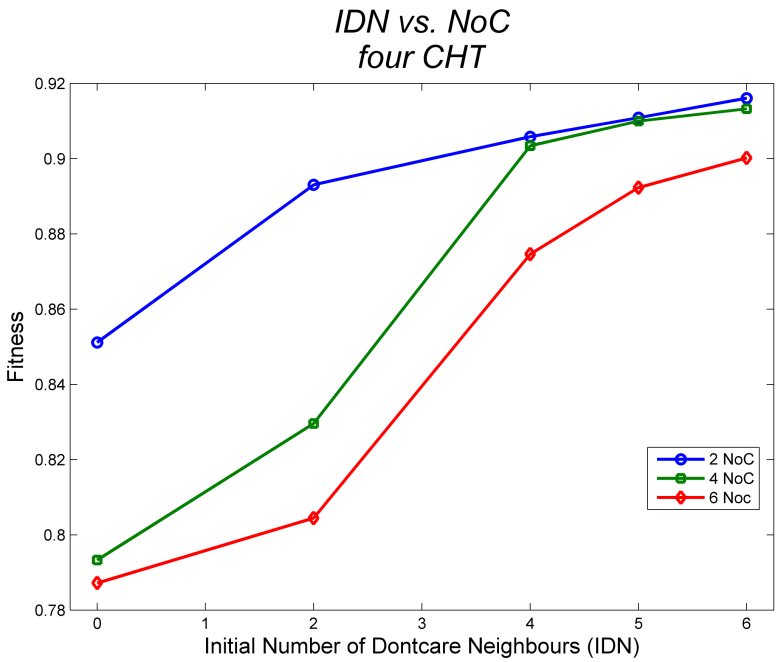


Figure 11.26: Plotting IDN vs. NoC for four chemical types (CHT) reveals a minor deviation.

The graph in figure 11.26 is a compound of results for DoS values of zero to three. This data can be broken down into graphs for each particular DoS. Two of these are shown in figures 11.27 and 11.28. Zero and one DoS – see figure 11.27 (only one DoS is shown), have the expected sudden increase in fitness, but for two and three DoS – see figure 11.28 (only three DoS is shown), the line for two colours is almost flat and actually achieves lower fitness than four colours for some values of IDN. There is no obvious explanation why this happens, but it may be that getting the perfect shape of the organism is easier when more colours are used as this gives more detailed information about the neighbourhood of each cell.

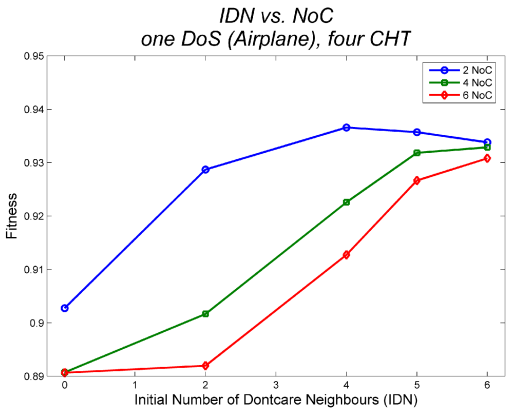


Figure 11.27: IDN vs. NoC for four chemical types (CHT) and one degree of symmetry (DoS) doesn't exhibit the effect found when using the average of DoS

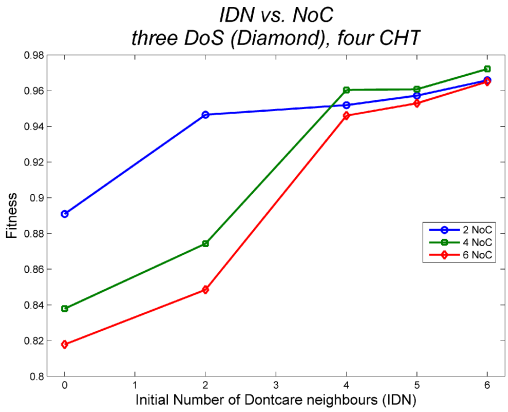


Figure 11.28: Two colours actually achieves lower fitness than four colours for IDN values of four to six in IDN vs. NoC with four chemical types (CHT) and three degrees of symmetry (DoS)

The graph for two DoS – see figure 11.29, closely resembles the one using averages, but deviates in that for five and six IDN the line for four colours lies below the one denoting six colours. This does not fit well with any assumptions made about the system, and no good explanation has been found. One likely cause is that this is just a result of random fluctuations. Taken into account that the experiments were only run twenty times each, and the deviation is small, this seems to be the most plausible explanation.

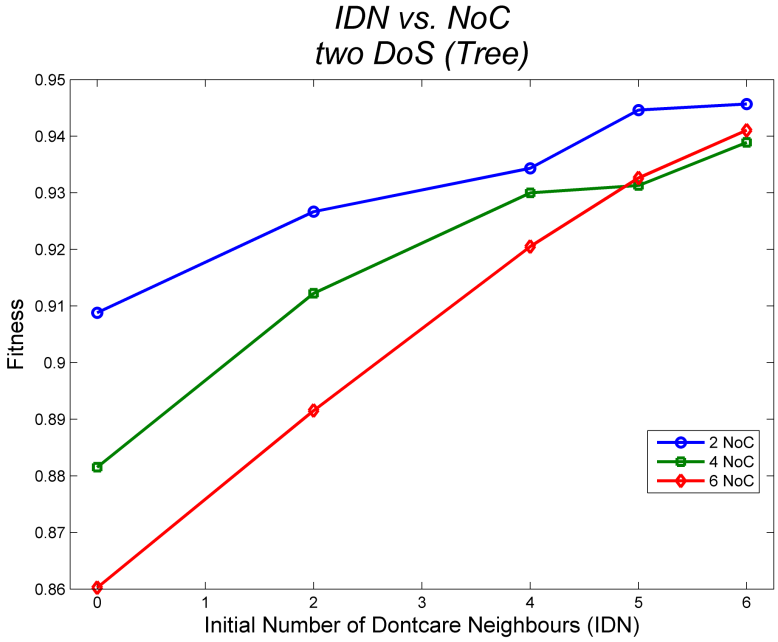


Figure 11.29: Six colours achieves higher fitness than four colours for IDN values of five and six in IDN vs. NoC for two degrees of symmetry (DoS)

Analysis taking the number of colours into account was also performed, but no new complication for the IDN-fitness correlation was found so the results will not be presented here. Although deviations do exists that may suggest a more complicated relationship between IDN and fitness, these deviations are small and most probably a result of random fluctuations.

Summary

Investigation of the effect of changing the initial number of don't care neighbours revealed a positive correlation between IDN and fitness when nothing was known about the other parameters. Further, it was found that the value of CHT has a great impact on this correlation. For certain values of CHT the correlation is not always positive. This means that when deciding the value of IDN, the value of CHT must also be taken into account. The degree of symmetry also has an impact on the IDN-fitness relationship as different values of DoS changes the strength of this relationship. The number of colours did, however, not have such an impact. In general, with increasing NoC, the fitness achieved decreases, but the relationship between IDN and fitness remains more or less the same.

11.3.2 The Number of Chemical Types

The second parameter to be investigated was the number of chemical types (CHT) – see chapter 9.3. Unfortunately, problems similar to those described in chapter 11.3.1 occur also here. Achieving growth may be an increasing problem with increasingly many chemical types. This is because with each extra chemical type it becomes increasingly difficult to get all

chemical concentrations above their given thresholds. The increased complexity and possibilities of the development process with increasing CHT makes the search more random, implying a more difficult search. Additionally, the search space is increased because of the extra parameters which need to evolve (one for each chemical type). This will lead to lower fitness score when all other parameters, like the number of generations to evolve, are held constant.

The value of CHT denotes the number of different chemical types used in the system. This number does not change during evolution, or even across different cells; if two chemical types are used, all proteins check the concentration of both these chemical types

The allowed values for CHT are all non-negative integers. CHT values of zero, one, two, four and eight was chosen to be used in this experiment series.

General effect

To find the general effect changing the value of CHT has on fitness, the data was plotted as a bar graph with standard deviations on top. The data was obtained using the highest fitness achieved in every run, and for every experiment. This was then organized in such a way that the various values of CHT could be plotted with their corresponding average and standard deviation.

The graph – see figure 11.30, shows that the average fitness is highest for low values of CHT and lowest for high values. The average fitness achieved decreases almost linearly with increasing CHT, with a high for zero CHT (0,90) and low for eight CHT (0,83). The standard deviation is relatively high for all values of CHT, but lower for low than high CHT. It increases from a low 0,08 at zero CHT to a high 0,11 at eight CHT. With respect to the range of average fitness values, the standard deviations are rather large, making it difficult to draw any statistical conclusion.

The impression given by this graph is that there exists a negative correlation between CHT and fitness. The fact that the decrease is almost linear indicates that the correlation is steady and robust. To test this impression for statistical significance, a one-way ANOVA test was employed to compare group means. The ANOVA test was chosen for reasons previously discussed – see chapter 11.3.1.

The results obtained from testing the data at a 95 % confidence interval strengthens the assumption that there exists a negative correlation between fitness and CHT. It shows that there are indeed statistically significant differences among the group means. When performing a more thourough investigation, however, the test shows that the difference between group means for zero and one CHT is not statistically significant, as there is a 24 % chance they are actually equal. This is a relatively high chance, but that alone is not enough to invalidate the hypothesis that a negative correlation does exist. After all, the average fitness achieved for one CHT *is* lower than that for zero CHT, it is just not statistically significant lower.

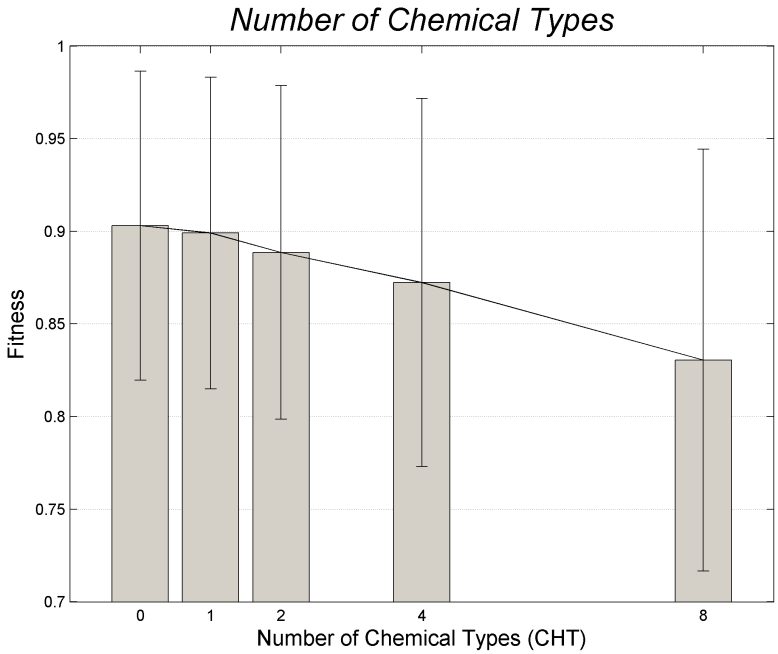


Figure 11.30: Fitness achieved using various numbers of chemical types (CHT)

According to the findings so far, the lowest CHT should be chosen to achieve highest possible fitness. As the lowest CHT value is zero, which corresponds to the development not using the concept of chemicals at all, does this mean that including chemicals in the development system was a bad choice? To answer this question, a more thorough analysis is necessary. The standard deviations are relatively high, and the other three parameters are not known. This means there might be other, maybe more complex, correlations between CHT and fitness, given various values for the other parameters.

To get a clearer picture of the relationship between CHT and fitness, three more graphs were created, each showing this relationship given each of the other three parameters. A quick visual inspection of these graphs revealed the one plotting CHT for each IDN to be the most interesting one. Therefore CHT vs. IDN was analysed first, followed by DoS and NoC.

Number of Chemical Types vs. Initial Number of Don't Care Neighbours

The graph plotting CHT vs. IDN – see figure 11.31, indicates that when taking the number of initial don't care neighbours into account, the relationship between the number of chemical types and fitness becomes more complex, just as it did for IDN when introducing CHT – see chapter 11.3.1.

For zero, four and five IDN the fitness is steadily decreasing with increasing CHT. The curve for zero IDN has a slight left-hook, while four and five IDN have a slight right-hook. Two and six IDN have an increase in fitness from zero to one CHT, followed by a decrease from one to eight CHT.

IDN values of zero, four an five do not reveal much new information on the relationship between CHT and fitness, they more or less show the same trend as was found in the bar graph of CHT, depicted in figure 11.30. For these values of IDN, not using chemicals at all seems to be the best option, as using chemicals only leads to a decrease in fitness, and the more chemical types used, the worse the fitness achieved. Two and six IDN, on the other hand, show a more interesting pattern. In these two cases it seems like including chemicals in the development process has a positive effect on fitness. However, using too many chemical types has a negative effect on fitness; when using more than one chemical type, the fitness decreases with increasing CHT. Another thing to notice is the curve for six IDN being flatter (fitness range is only 0,03) than for the other four (next lowest fitness range is 0,06). This indicates that, relative to the other values of IDN, using six IDN reduces the effect the value of CHT has on fitness.

The explanation for the peak at one CHT for six IDN is most likely as straight forward as this: When the neighbourhood is not used as a guideline for development, something else is needed. If no guidelines are present, it would be very difficult to develop regular shapes, as opposed to just a blob of cells. It is not impossible, however, to develop regular shapes even without taking neighbourhood into consideration, or by using chemicals; by carefully selecting the proteins in the zygote (initial cell) basic shaping is possible. In addition, six IDN does not mean that neighbourhood is not taken into consideration, only that it is initially ignored in the evolution. But, as was seen in figure 11.17, six IDN on average results in about five don't cares. Even though five don't cares does give development some guiding, it is reasonable to assume this is not enough. It also seems reasonable that starting with just a hint of guiding (i.e. using one chemical type) will make it easier for evolution to evolve more fit individuals than with no guiding at all.

The peak displayed by two IDN at one CHT, is not as easy to explain. Using one-way ANOVA to test for statistical significant difference between group means for zero and one CHT, shows that the peak may actually be just a random fluctuation, as there is a 24,45 % chance there is no real difference between these two groups. Although the ANOVA test fails to confirm the visual impression of a peak at one CHT, a visualization of the data where CHT vs. IDN is plotted for each value of NoC and DoS, shows that this trend is present in roughly half of the plots, indicating this could be an interesting finding after all. The fact that a similar deviation was found in the graph plotting IDN vs CHT – see figure 11.14, strengthens the assumption of this as an interesting finding. It might be that the combination of two IDN and one CHT is a particularly powerful one, but with only the data obtained from these experiments it is difficult to draw any conclusions. Further investigation into this issue by running some carefully planned experiments may be useful to get a deeper understanding of how the development system works.

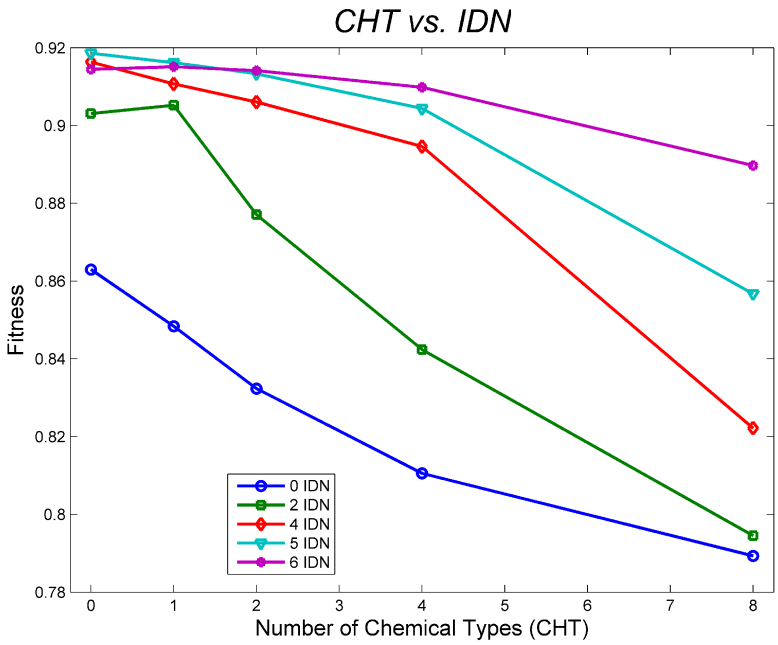


Figure 11.31: Fitness achieved using various numbers of chemical types (CHT) and initial number of don't care neighbours (IDN)

The lines for two and six IDN runs parallel for zero to one CHT. The values two and six are both roughly equally far apart from the optimum of just above four, which in chapter 11.3.1 there was found support for being the optimal number of don't care. In addition to this, four and five IDN runs in parallel. This makes it tempting to hypothesize that two IDN could be parallel to six IDN for all values of CHT, if they were allowed to evolve for more generations. After all, increasing CHT increases the search space, and hence may require more generations to achieve good fitness. However, no support for this hypothesis was found when analysing the fitness during evolution for the best individual in each generation; all tested values of CHT shows the same pattern of fast increase in the beginning and then levelling off – see figure 11.32.

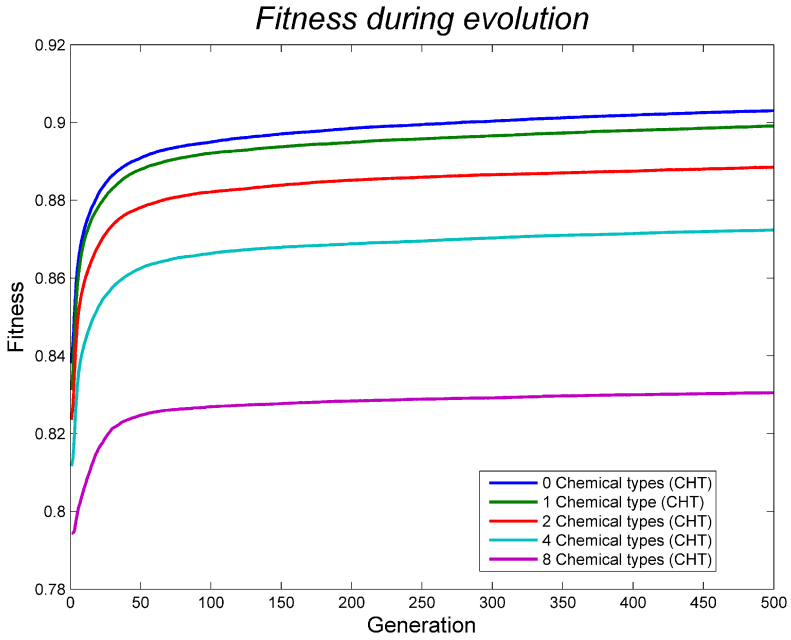


Figure 11.32: Change in fitness during evolution when using various numbers of chemical types (CHT)

Number of Chemical Types vs. Degrees of Symmetry

When plotting CHT vs. DoS – see figure 11.33, a graph strikingly similar to the one plotting IDN vs. DoS – see figure 11.24, appears. The main difference is that this one is a mirror image of the other, with sinking curves instead of rising ones. Thinking about it, this is not such a surprise, as both the value of CHT and IDN have an effect on the expressibility of the development system. The expressibility is in turn assumed to be in close connection to the degree of symmetry of the target. Of course, IDN does not have a direct effect on the expressibility, as any number of don't care neighbours may result from any given IDN. But FDN affects expressibility, and FDN is greatly influenced by IDN, as was shown in chapter 11.3.1.

The figure shows four close-to linear lines, all monotonically decreasing. Three of the lines (one, two and three DoS) are clustered together in the upper half of the graph, while the last line (zero DoS) lies in the lower part of the graph, relatively far below the other three. The slope of the lines varies to some extent: the steepest slope is found in zero DoS, followed by three, two and one DoS, the latter being nearly flat.

As the achieved fitness itself is not of much interest in this analysis, the fact that zero DoS achieves so much lower fitness than the other three is irrelevant; it is the shape and the slope of the lines that are the interesting aspects. The fact that all the lines are almost linear indicates a steady and predictable relationship between CHT and fitness, regardless of DoS. The slope of the lines gives an indication of the strength of the relationship between CHT and fitness; the steeper the slope, the stronger the relationship. According to this graph, it seems that as the

degree of symmetry decreases, so does the strength of this relationship. One line, however, does not fit into this hypothesis; zero DoS has the steepest slope of them all. In chapter 11.3.1 it was questioned whether the shape chosen to represent zero DoS really captured the essence of an asymmetric shape. It was also suggested that the shape chosen may actually be more like a shape with three DoS than one with zero DoS. If this is correct, the line for zero DoS may be disregarded and hence the hypothesis that the relationship between CHT and fitness is weakened with decreasing DoS, remains valid.

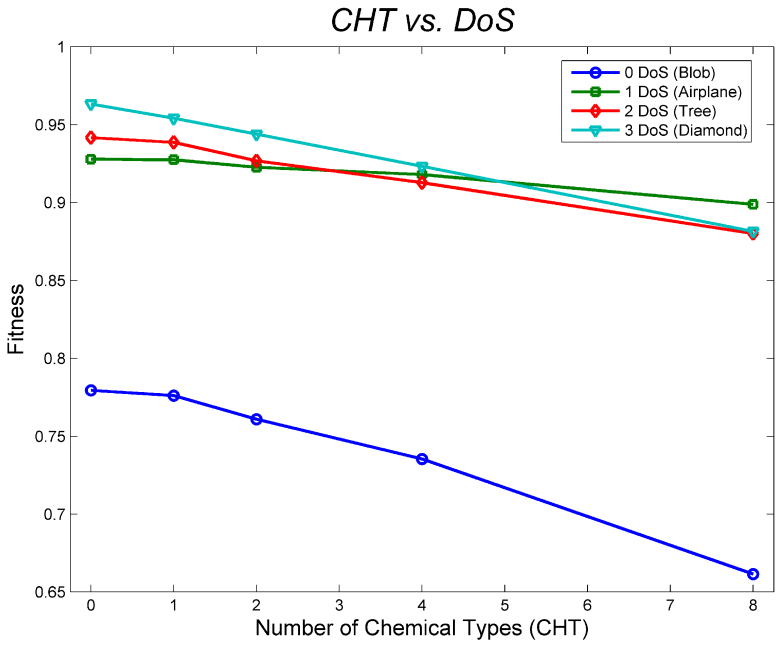


Figure 11.33: Fitness achieved using various numbers of chemical types (CHT) and degrees of symmetry (DoS)

CHT vs. DoS was also plotted for each value of IDN and NoC to investigate closer the components of the graph in figure 11.33. Looking first at the graphs for each IDN, the same shape of the curves occurs as in the plot showing CHT vs IDN – see figure 11.31. This is as expected, and indicates that the shapes of the curves, as found in CHT vs IDN, is representative regardless of the value of DoS. Further, it is found that the curve for three DoS shifts upwards, and one DoS shifts downwards, relative to two DoS, with increasing IDN. This is an indication that, relative to two DoS, one DoS achieves higher fitness for low values of IDN, and three DoS achieves higher fitness for high values of IDN. When looking at CHT vs DoS for each value of NoC, the same shifting-pattern is found, only inverse; three DoS shifts down, and one DoS shifts up with increasing NoC. However, the effect is much less obvious in these plots. This shifting-effect is illustrated in figure 11.34.

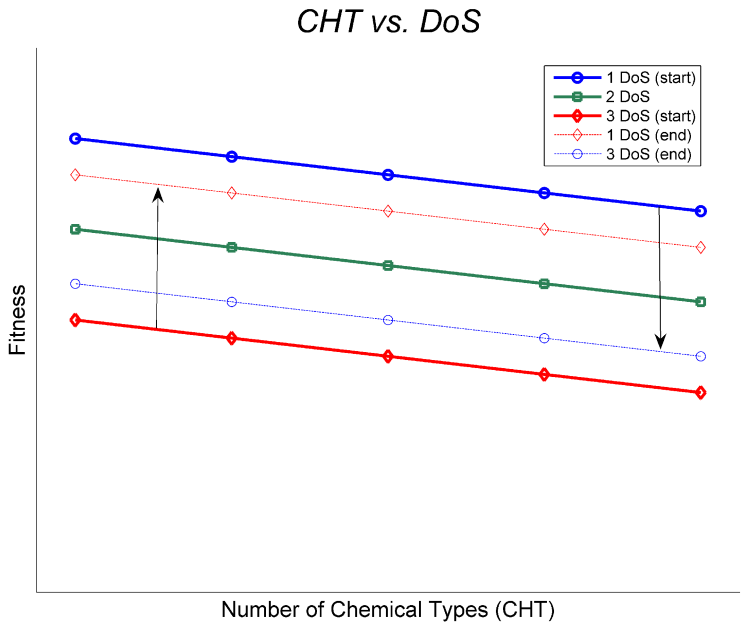


Figure 11.34: The shifting of one and three DoS relative to two DoS, with increasing IDN, and to some extent with decreasing NoC

When plotting CHT vs. DoS for each combination of IDN and NoC, it is found that the graphs plotting averages give a good impression of the underlying values. Some exceptions exist, like bimodal curves – see figure 11.35, curves with a peak at one or two CHT – see figure 11.36 and 11.37, and plots where the curve order is upside-down – see figure 11.38. Testing the bimodal curves for statistical significance shows that the fitness values between the two peaks can not be said to be statistically lower than the fitness at the peaks. This indicates that the bimodal shape of the curves is most likely just a random fluctuation. The close-to flat curves indicate that changing the value of CHT has little effect on the fitness. A peak for one CHT is found in several graphs, and fits well with previous findings – see figure 11.31. The plot showing a peak for two CHT for zero DoS, in addition shows that four CHT achieves higher fitness that both zero and one CHT. This indicates that for zero DoS and the combination of five IDN and two NoC, more chemical types are better than few. However, the peak at two CHT is not significantly different from the fitness achieved for other values of CHT, so it may be no more than a random fluctuation. The upside-down ordering of the curves occurs for values of IDN and NoC that makes it particularly difficult to develop anything with high fitness (zero IDN and six NoC). Under these circumstances it seems that shapes of fewer degrees of symmetry are easier to evolve, as one DoS achieves the highest fitness for all values of CHT followed by two and three DoS. Zero DoS achieves the lowest fitness of all, but as earlier explained, the shape chosen for zero DoS may not actually represent zero degrees of symmetry very well, and hence may be disregarded.

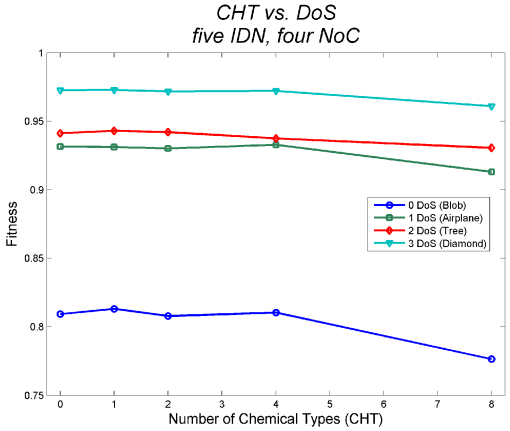


Figure 11.35: Bimodal curves found for zero, one and three DoS when plotting CHT vs. DoS for five IDN and four NoC

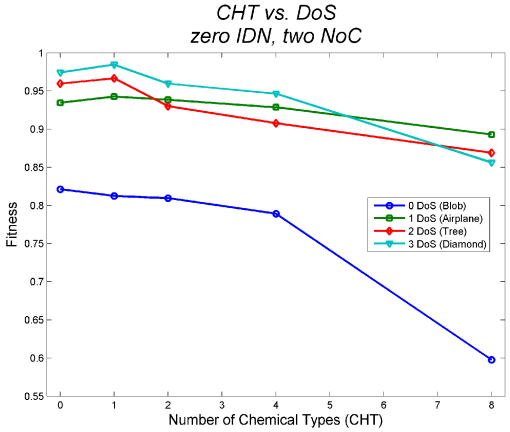


Figure 11.36: A peak at one CHT for one, two and three DoS was found when plotting CHT vs. DoS for zero IDN and two NoC

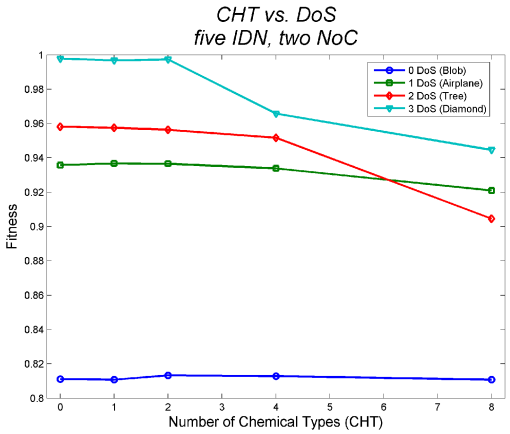


Figure 11.37: A peak at two CHT for zero DoS was found when plotting CHT vs. DoS for five IDN and two NoC

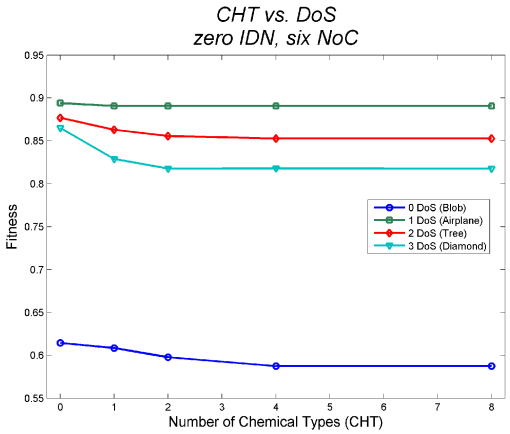


Figure 11.38: For CHT vs. DoS for zero IDN and six NoC, one DoS achieves highest fitness for all values of CHT followed by two, three and finally zero DoS

Number of Chemical Types vs. Number of Colours

As with CHT vs DoS, plotting CHT vs. NoC – see figure 11.39, results in a mirror image of the plot showing IDN vs. NoC – see figure 11.25. This is as expected, following a similar reasoning as the one used in chapter 11.3.2.

The figure shows three close-to linear lines, all monotonically decreasing, and with a more-or-less equal distance from each other. The three lines runs in almost perfect parallel.

Since the three lines are parallel to each other, with equal distance between them, it seems like the effect of changing the number of colours is to shift the fitness a given amount up or down

with decreasing and increasing number of chemical types. It seems like no additional complexity in the relationship between the number of chemical types and fitness emerges when changing the number of colours.

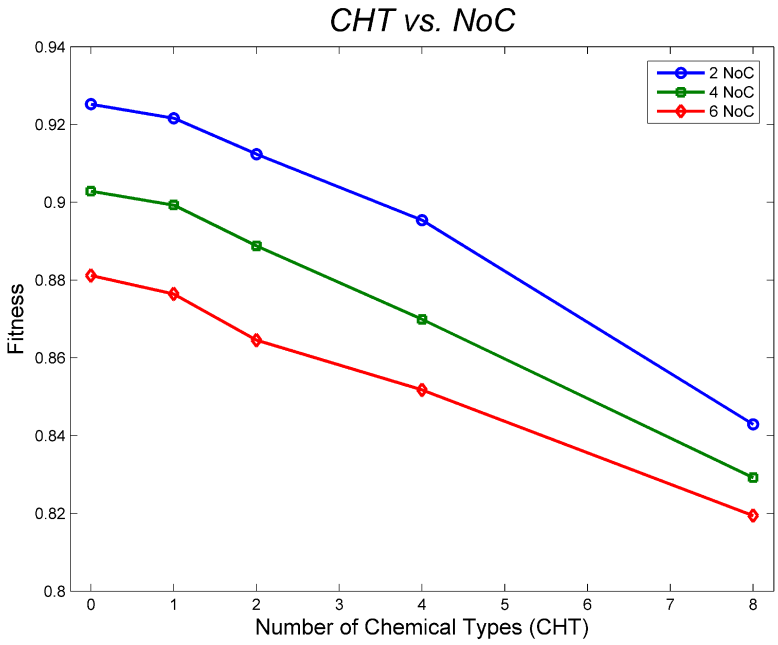


Figure 11.39: Fitness achieved using various numbers of chemical types (CHT) and number of colours (NoC)

Also when plotting CHT vs. NoC for each of IDN and DoS, the same pattern appears. For CHT vs. NoC with various values of IDN, some minor deviations are found. The shape of the curves are as expected when comparing to the plot of CHT vs. IDN – see figure 11.31, and they run more-or-less parallel to each other. However, for two IDN – see figure 11.40, the curves for two and four colours have a peak at one CHT while six colours has its peak at zero. The differences between the fitness at zero and one CHT for all values of NoC are too small to have statistical significance, so this minor deviation is not investigated further.

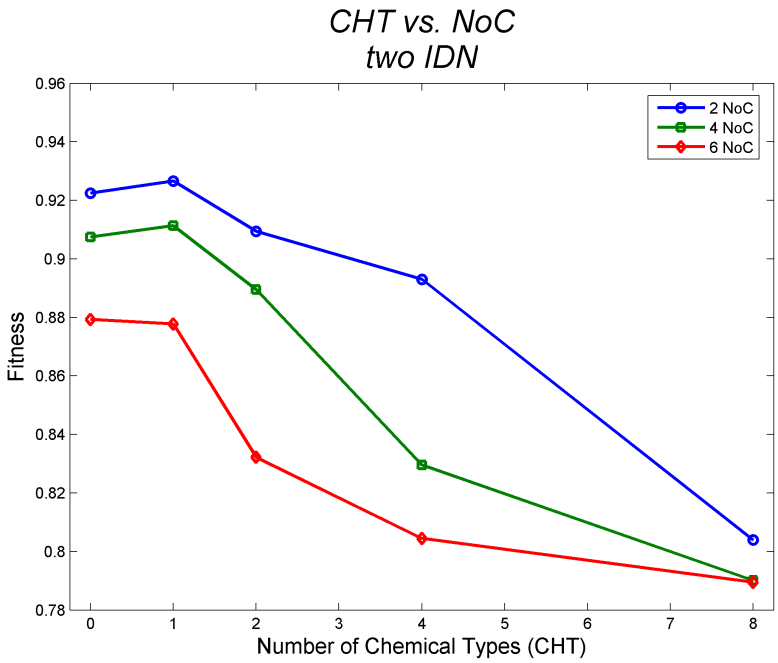


Figure 11.40: Peak at one CHT for two and four NoC in combination with two IDN

For four, five and six IDN, a more interesting deviation appears – see figure 11.41. The curves for four and six NoC are as expected with respect to the analysis already performed. Two colours has a steeper slope, resulting in two and four colours achieving an almost equally high fitness at four CHT. This indicates that when using two colours, the effect on fitness when changing CHT is greater than with four or six colours for four, five and six IDN. There is no obvious explanation for this phenomenon, and the difference between two colours and four or six colours is not that large, indicating it is probably not significant. In addition, using few colours is an easier task for the genetic algorithm, which makes this finding a contradiction to the finding in chapter 11.3.2. There, it was found that the number of chemical types used has a greater effect on fitness when evolving a more complex shape. This strengthens the assumption of this being merely a random fluctuation.

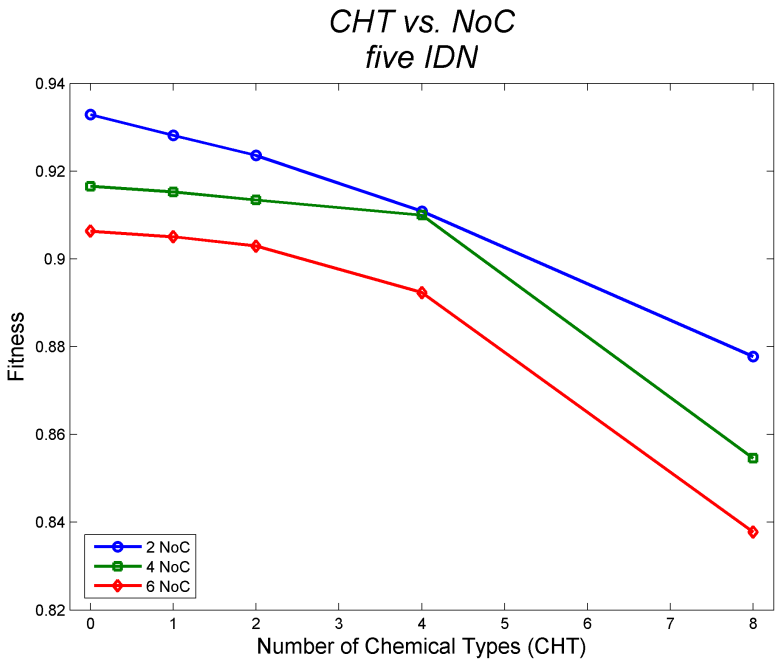


Figure 11.41: The curve for four NoC is not parallel to those of two and six NoC for five IDN

When investigating CHT vs. NoC for various values of DoS, no noteworthy deviations are found: they all highly resemble the plot of CHT vs. NoC using averages.

Although the plot of CHT vs. NoC using averages seems to be very representative for the data for specific parameter values, there are some interesting deviations. For the combination two DoS and two IDN, all three values for NoC have a peak at one CHT – see figure 11.42. However, the curves retain their properties from the average plot of being parallel and equally shaped. This indicates that for these parameter values, it is not the number of colours that determines that one CHT is best, it is the combination of two DoS and two IDN.

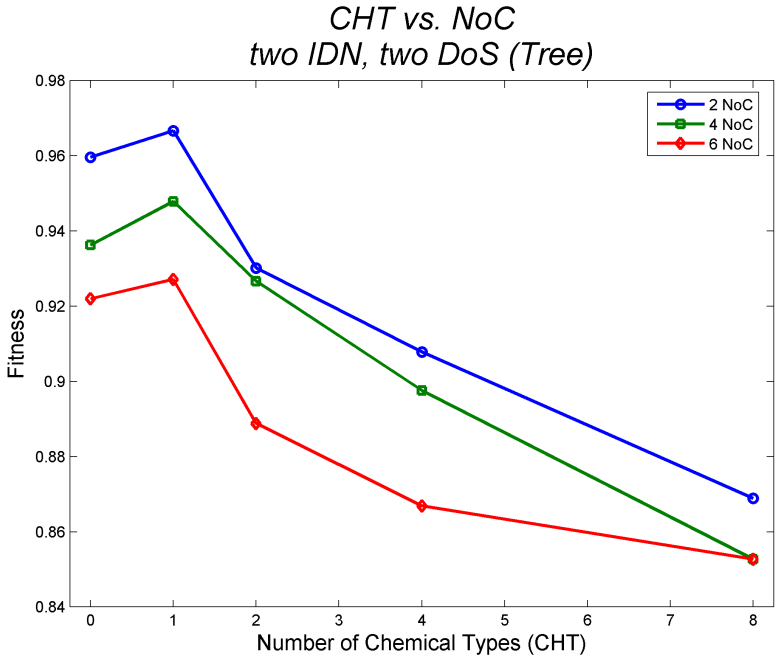


Figure 11.42: The curves for all NoC have a peak at one CHT for the combination of two DoS (tree) and two IDN

For the parameter combination two DoS and five IDN, - see figure 11.43, the curves for four and six colours are almost on top of each other while two NoC seems to be completely unrelated the the others, starting high above at zero CHT, then going below at four CHT, and ending a little above at eight CHT. Two questions arise here: why is there no difference between four and six NoC? Why is the curve for two NoC so different from those of four and six?

The most plausible explanation for the first question is that the shape and placement of the colours, for four and six colours, makes it easy for both to achieve equally high fitness for five IDN. In other words; it is not as much a property of the development system, but of the fitness function used in the genetic algorithm that causes this to happen.

Thinking about it, the second question isn't that much of a mystery after all. The only thing that is out of place is the fitness achieved at four CHT. And a deviation at four CHT was also found in the investigation of CHT vs. NoC for each IDN. Keeping to what was found there, this can also be regarded as just a random fluctuation.

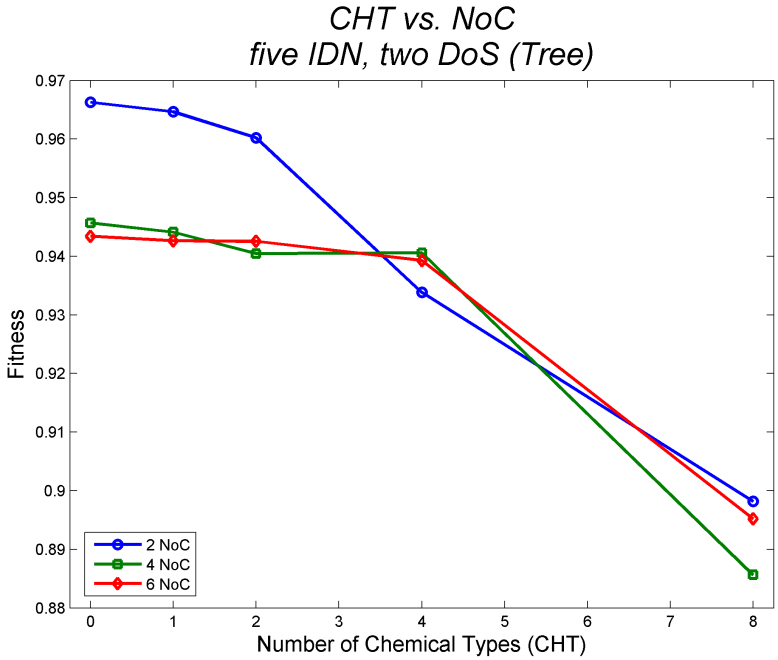


Figure 11.43: The curves for four and six NoC are almost indiscriminant from each other, and the curve for two NoC does not follow their pattern, for the combination of two DoS (tree) and five IDN.

For the combination of one DoS and four IDN – see figure 11.44, a bimodal curve was found for two NoC. The fitness for two NoC decreases from zero to one CHT, then increases from one to two CHT where it reaches its peak, and then decreases from two to eight. Four NoC increases from zero to one CHT, and then decreases, while six NoC decreases all the way from zero to eight CHT. The bimodal shape of the curve for two NoC is not statistically significant. However, it is interesting to note that the highest fitness is found at decreasing CHT with increasing NoC. When few colours are used, the genetic algorithm may focus more on evolving the correct shape. This may be a small indication that chemicals are important for the shaping of the organism.

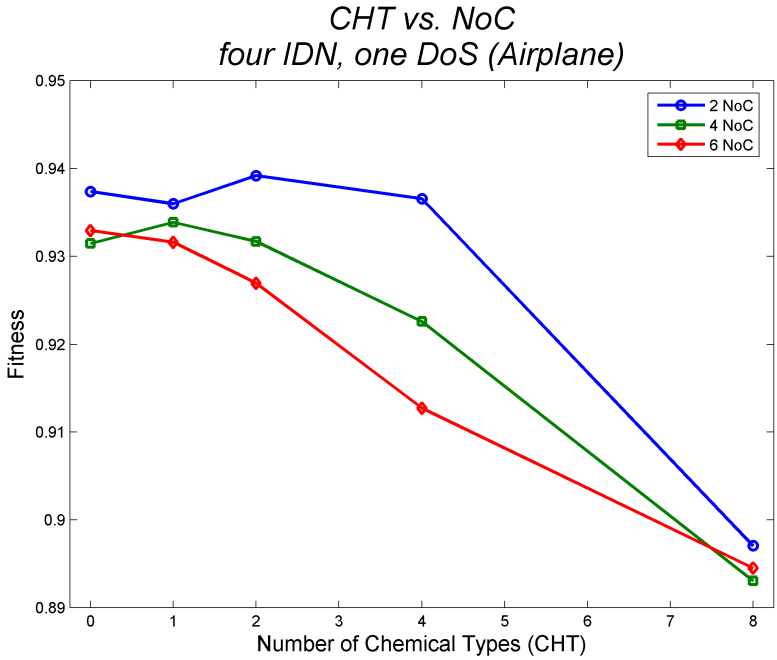


Figure 11.44: Different peaks for different values of NoC, for the combination of one DOS (airplane) and four IDN

Summary

The investigation of the effect changing the number of chemical types (CHT) has on fitness, showed that, when disregarding other parameters, there exists a negative correlation between CHT and fitness. When taking other parameters into account, it was found that the relationship is not all that simple. For certain values of initial numbers of don't care neighbours (IDN) the correlation is not always negative, meaning the highest fitness is not always achieved when not using any chemical types. Because of this, it is important that the value of IDN is taken into account when deciding the value to use for CHT. The degree of symmetry (DoS) also has an impact on the relationship between CHT and fitness: Different values of DoS changes the strength of this relationship, and some support for the hypothesis that the strength of the relationship between CHT and fitness increases with increasing symmetry, was found. The number of colours (NoC) did, however, not have such an impact on this relationship. In general, with increasing NoC, the fitness achieved decreases, but the relationship between CHT and fitness remains more or less the same.

11.3.3 The Degree of Symmetry

The third parameter to be investigated was the degree of symmetry (DoS) of the target shape. Because of the way the development system is built to work, it is plausible to assume that the degree of symmetry in the target will have an effect on how easy it is to develop. The symmetry of a shape is divided into four categories: zero, one, two and three degrees of symmetry. A shape has three degrees of symmetry if it is symmetric in all three dimensions

right-left, top-bottom, and front-back. An example of such a shape is a diamond. A shape is considered to be of two degrees of symmetry if it is symmetric in two of these dimensions, and of one degree of symmetry if symmetric in only one dimension. Shapes which are completely asymmetric fall into the last category, zero degrees of symmetry.

The biggest problem when testing this parameter was to find a representative shape for each of the four categories. After some consideration, the choice fell on the following four shapes: a blob, an airplane, a tree, and a diamond, for zero, one, two and three DoS respectively.

Because of the way the fitness function evaluates the phenotype, and because of the specific shapes chosen, a direct comparison of the fitness for each of the four values of DoS would be misleading. Even if two shapes achieve equally high fitness, it doesn't mean they resemble their respective target equally much. This is, however, not regarded as a big problem, as a direct comparison of the fitness is not that interesting anyway.

The blob chosen to represent zero DoS may have been a bad choice, as it turns out. Because it essentially is a cube, which is symmetric in three dimensions, embedded with small blobs, it is likely to be treated as a shape with three DoS by the fitness function. This happens because the shape is greatly dominated by the cube shape, and if the cube is developed the shape will be assigned high fitness even if the blobs are in the wrong place. This is supported by the findings that zero DoS achieves comparable fitness with three DoS – see 11.3.1 and 11.3.2. Because of this, the results obtained for zero DoS will be given less attention than the results from the other three.

General effect

To find the general effect changing DoS has on fitness, the data was plotted as a bar graph with standard deviations on top.

The graph – see figure 11.45, shows that the average fitness achieved is lowest for zero DoS (asymmetric), then increasing with increasing DoS with the highest average fitness achieved for three DoS. The difference in fitness between zero DoS (0,74) and the other three (0,92 – 0,93) is noteworthy. Also, the standard deviations display relatively large variations, from 0,02 at one DoS to 0,09 at zero DoS.

The impression given by this graph is that there exists a positive correlation between DoS and fitness. But the relationship is far from linear, indicating the relationship is probably not a straightforward one. This is as expected, because of the way the fitness function evaluates the fitness of a shape, as discussed in chapter 11.3.3.

Even though a direct comparison of the fitness will most likely be misleading, it is interesting do test for statistically significant differences, as this could give some hints about the relationship between DoS and fitness. Once again, a one-way ANOVA test were employed to test for statistical significance. The ANOVA test was chosen for reasons previously discussed – see chapter 11.3.1.

At a 95% confidence interval, it was shown that both zero and three DoS are significantly different from all the others, while one and two DoS are so close to each other that it cannot be ruled out their fitness is actually equal.

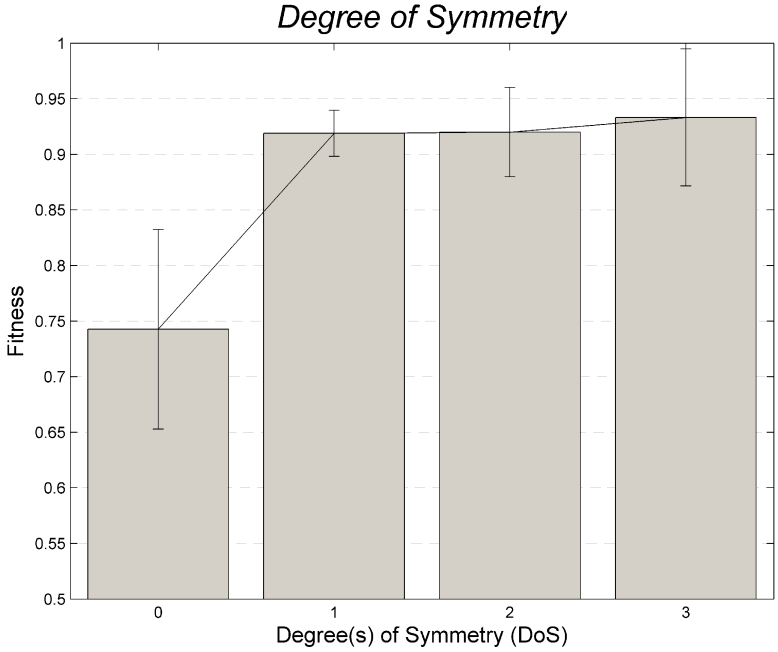


Figure 11.45.: Fitness achieved with various degrees of symmetry (DoS) in the target

To get more comparable fitness values, the fitness for each of the best evolved phenotypes were recalculated using a fitness function which reduces the unfair differences in fitness based on the size of the target (the number of cells in the target). The fitness function used during the evolution compares each point in the target with the corresponding point in the evolved phenotype. Fitness increases with each matching point. This approach assigns phenotypes which are compared to small targets with relatively higher fitness than to those compared to bigger targets.

The fitness function used to rescale fitness based on the size on the target, on the other hand, counts both how many correct non-cell points and how many correct points containing a cell there are. This makes the fitness less dependent on the number of cells in the target.

As can be seen in figure 11.46, the rescaled fitness is to some extent very different from the "normal" fitness – see figure 11.45. Now the fitness score for one DoS is the lowest one, while zero DoS is slightly lower than two DoS. Also, there is a bigger and more linear increase in fitness from one DoS to tree DoS compared to figure 11.45. The fact that the fitness achieved for zero DoS is higher than that for one DoS, fits well with the assumption made previously that the blob used to represent zero DoS actually resembles a shape of tree DoS. An explanation for the relatively low fitness for the blob compared to tree DoS, may be that if it

really is evolving as a cube, then the failure to evolve the blobs embedded in the cube will results in a loss of fitness.

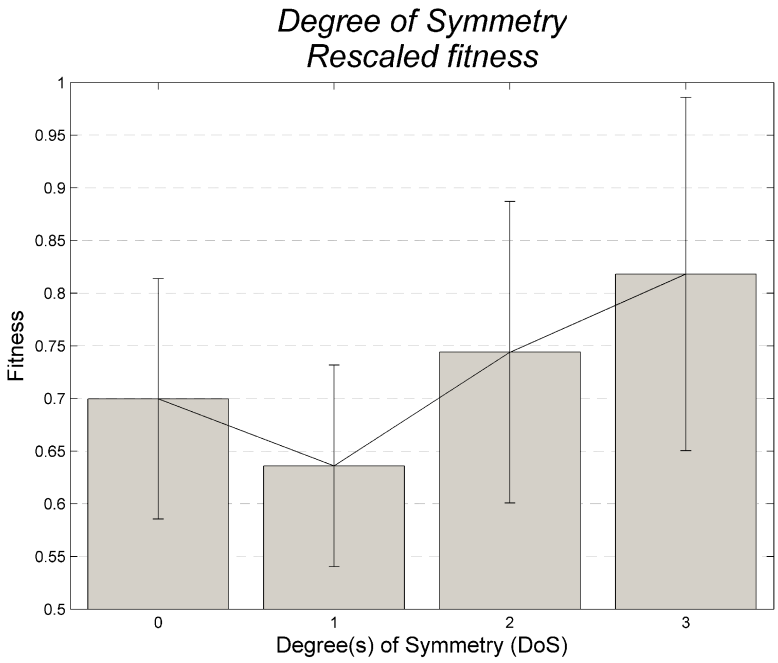


Figure 11.46.: Fitness achieved with various degrees of symmetry (DoS) in the target when the fitness is rescaled to be more comparable

Even though the rescaled fitness values seem to fit better with previous findings, and make the fitness more comparable, it is important to remember that the fitness function used to get these values is not the same as the one used by the genetic algorithm during evolution. The following analysis will only make use of the fitness values obtained using the "normal" fitness function.

According to the findings so far, the development system will perform better when evolving and developing a symmetrical target than a more asymmetric one. However, the relationship appears to be complex, and it is hoped that looking at the data with fixed values of the other three parameter will give a deeper understanding of this relationship. In addition, the question of wether the shapes used in the experiments represent the various degrees of symmetry good enough, still remains.

The graphs where DoS is plotted against each of the other three parameters resemble each other to a great extent. A quick visual inspection of each of them shows that none of them seem more interesting than the others, Therefore, the same pattern of analysis as in previous investigations will be applied. DoS vs. initial number of don't care neighbours (IDN) will be analysed first, followed by DoS vs. number of chemical types (CHT), and finally DoS vs. number of colours (NoC).

Degrees of Symmetry vs. Initial Number of Don't Care Neighbours

The graph plotting DoS vs. IDN – see figure 11.47, does not display any large deviations from the bar graph – see figure 11.45. All values of IDN have an increase in fitness from zero DoS to one DoS, followed by a small increase from one to two DoS for four, five and six IDN, and a small decrease for zero and two IDN. Finally, from two to tree DoS, there is a bigger increase for all value of IDN except zero, which has a further slight decrease. This graph also gives a good representation of the variance seen in figure 11.45, with smallest variance at one DoS, increasing variance up to three DoS, and the biggest variance at zero DoS.

The coherence between the standard deviations in figure 11.45 and the variance in fitness achieved for the different values of IDN makes it plausible to assume that most of the variance can be accounted for when looking at the value of IDN. The differences between the fitness achieved at one DoS is relatively small (only 0,03), indicating that for this value of DoS the value chosen for IDN doesn't have much effect. For two and three DoS, the differences are greater, indicating the value of IDN has a greater effect. This could be an indication that with increasing symmetry the value of IDN becomes more and more important. However, for zero DoS this hypothesis doesn't hold: here the differences is even bigger than for three DoS. But, as discussed in chapter 11.3.3, this could be because the blob chosen to represent zero DoS actually is evaluated as a shape with three degrees of symmetry by the fitness function.

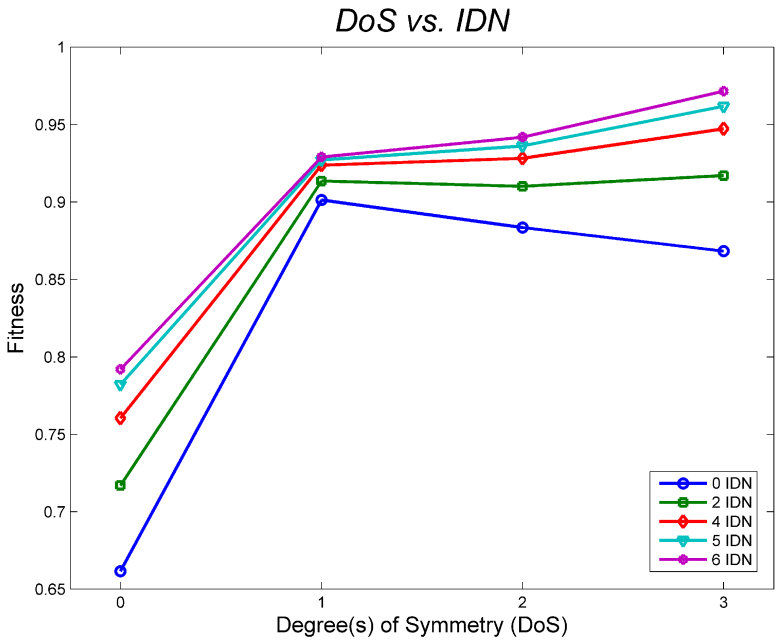


Figure 11.47.: Fitness achieved with various degrees of symmetry (DoS) and intial number of don't care neighbours (IDN)

Degrees of Symmetry vs. Number of Chemical Types

The graph showing DoS vs. CHT – see figure 11.48, is highly similar to the one showing DoS vs. IDN – see figure 11.47. Here too the lines seem to originate from more or less the same spot at one DoS, and spread out in both directions.

This graph can therefore be interpreted analogously to the one of DoS vs. IDN. It seems that the importance of the CHT value increases with increasing DoS. Here too the exception is for zero DoS, which looks more like three DoS.

Taken together with the findings in chapter 11.3.3 it seems plausible to believe that the reason for the small differences in fitness at one DoS, is that this shape is harder to develop, making the effect of changing the value of CHT and IDN almost insignificant.

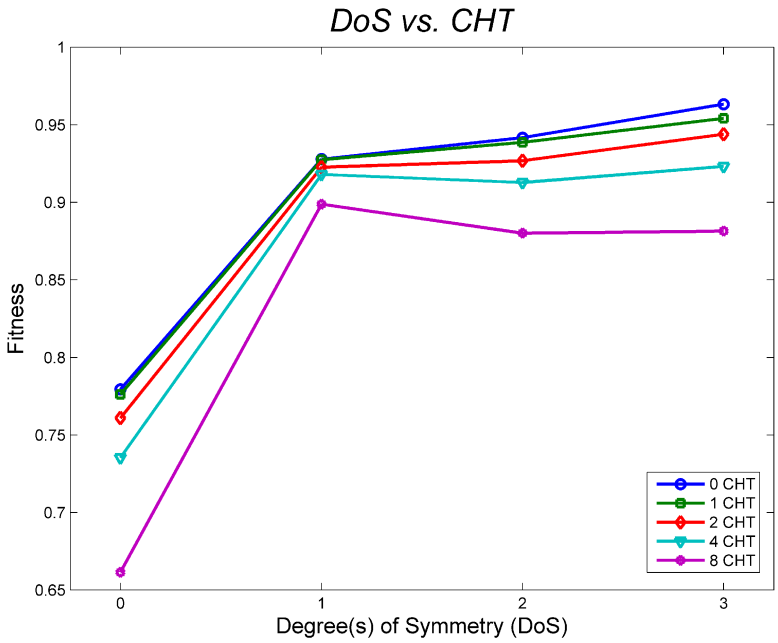


Figure 11.48: Fitness achieved with various degrees of symmetry (DoS) and number of chemical types (CHT)

Degrees of Symmetry vs. Number of Colours

Also the graph showing DoS vs. NoC – see figure 11.49, is highly similar to both DoS vs. IDN and DoS vs. CHT – see figure 11.47 and 11.48. Once again the lines seem to originate from more or less the same spot at one DoS, and spread out in both directions. However, the spreading is less pronounced here than in the two other graphs.

Here also, it seems that the importance of the NoC value increases with increasing DoS, with the exception of zero DoS.

The hypothesis made above, that the shape for one DoS is harder to develop, seems to apply here also, although these three lines run more in parallel compared to the other two graphs.

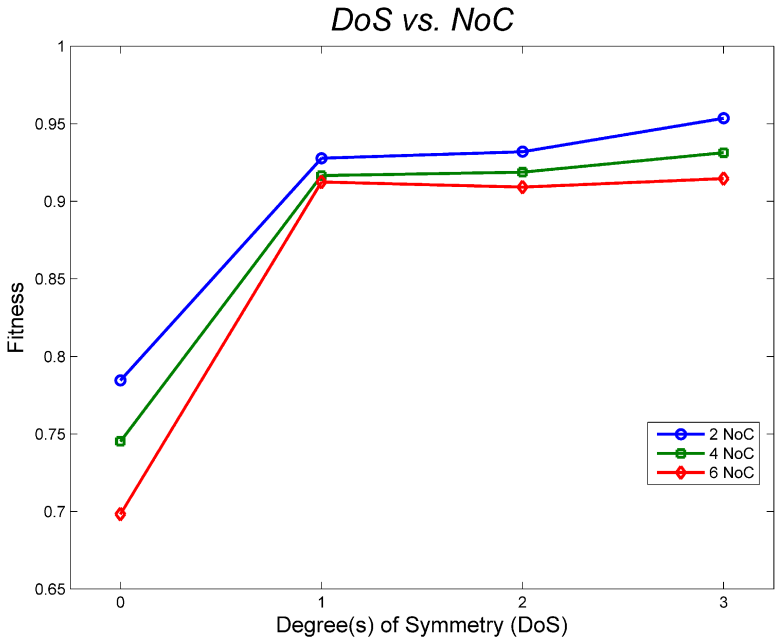


Figure 11.49: Fitness achieved with various degrees of symmetry (DoS) and number of colours (NoC)

Summary

The investigation of how well the development system is at developing (and evolving) shapes with different degrees of symmetry, indicated a positive correlation between the degree of symmetry and fitness when disregarding other parameters. Because a direct comparison of fitness may be misleading, both because of the way the fitness function calculates the fitness and because the choice of shapes to represent each degree of symmetry is questionable, one must be careful not to put too much meaning into these findings. Although both this and the increasing differences between fitness with increasing degrees of symmetry seem to support the hypothesis that the development system is better at developing symmetrical shapes, a more extensive experiment must be performed on this subject before any conclusions may be drawn.

11.3.4 Number of Colours

The fourth, and final, parameter to be investigated was the number of colours (NoC) used in the target organism. What the value of NoC actually denotes is the number of different cell types that may be present in the organism. However, as all that separates the different cell types, at least in the experiments performed in this thesis, is the colour of the cell, it is just as correct and also more convenient to discuss it as colours.

Changing the number of cell types ArtDev3D uses changes the search space. This is because increasing/decreasing the number of cell types increases/decreases the number of bits required to represent the parameter for cell type. As an increase in search space results in lower fitness score when all other parameters are held constant, an decrease in fitness with increasing NoC are to be expected.

A choice was made to run experiments with two, four and six NoC. As the number of colours was expected to only have a minor effect on the fitness, this was deemed enough to get an idea of its effect on fitness.

General effect

To find the general effect changing NoC has on fitness, the data was plotted as a bar graph with standard deviations on top. The data were obtained using the highest fitness achieved in every run, and for every experiment. This was then organized in such a way that the various values of NoC could be plotted with their corresponding average and standard deviation.

The graph – see figure 11.50, shows that the highest average fitness is achieved at two NoC. The fitness then decreases with increasing NoC, and the lowest average fitness is achieved at six NoC. The decrease in fitness with increasing NoC seems to be almost linear. However, as the graph lines are only made up of three data points, care must be taken not to put to much importance into this observation.

The impression given by the graph is that there exists a negative correlation between NoC and fitness. Because the decrease is almost linear, this indicates the correlation is steady and robust. To test this impression for statistical significance, a one-way ANOVA test were employed to compare group means. The ANOVA test were chosen for reasons already discussed – see chapter 11.3.1.

The results obtained from testing the data at 95 % confidence interval strengthens the hypothesis that there exists a negative correlation between fitness and NoC. The test shows that there are indeed statistically significant differences among the group means. However, this does not mean that a negative correlation has been proven, only that changing NoC has an effect on fitness. The graph is used as a visual aid to get an idea of the direction of this effect.

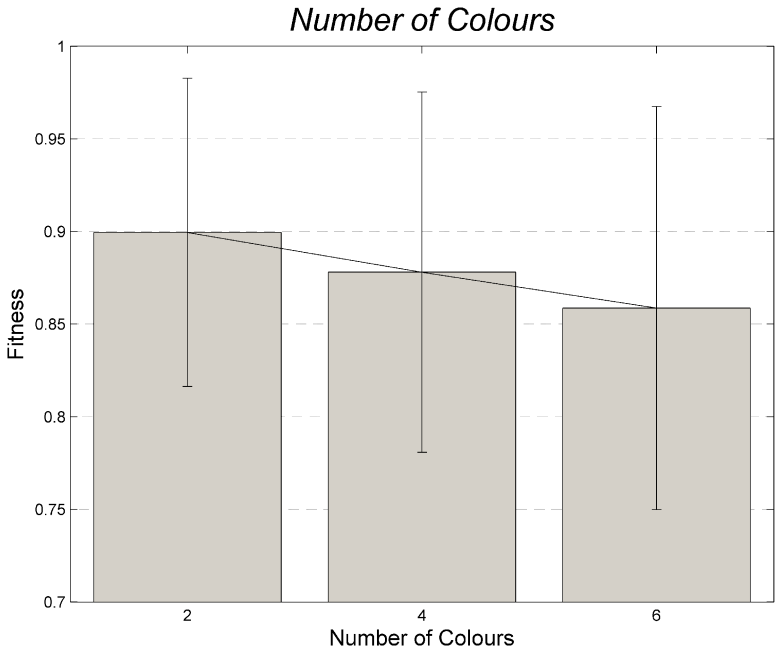


Figure 11.50: Average fitness achieved with standard deviations for various number of colours (NoC) in the target phenotype

As was done with the previous parameters, this one was also plotted against the other three to check for underlying relationships. The same pattern of analysis as before was applied: NoC vs. IDN will be analysed first, followed by NoC vs. CHT, and finally NoC vs. DoS.

Number of Colours vs. Initial Number of Don't Care Neighbours

The plot of NoC vs. IDN – see figure 11.51, shows how the effect of changing NoC is dependent on the value of IDN. All lines are nearly linear, except for zero IDN which has a slight left hook. The steepness of the slope of the lines seems to decrease with increasing IDN and, at the same time, the lines shift upwards. This indicates that with increasing IDN, the effect on fitness when changing NoC becomes smaller and, when NoC is held constant, the achieved fitness increases. The latter is the same effect as was found in chapter 11.3.1.

The former of these two effects fits well with the previous finding that, in general, fitness increases with increasing IDN – see chapter 11.3.1. This was hypothesised to be because increasing IDN led to an easier search for evolution, and increasing NoC complicates the search, because it increases the search space. It is expected that further complicating a complex search will lead to greater loss in fitness. The slight left hook at zero IDN is difficult to explain, but is most likely just a random fluctuation.

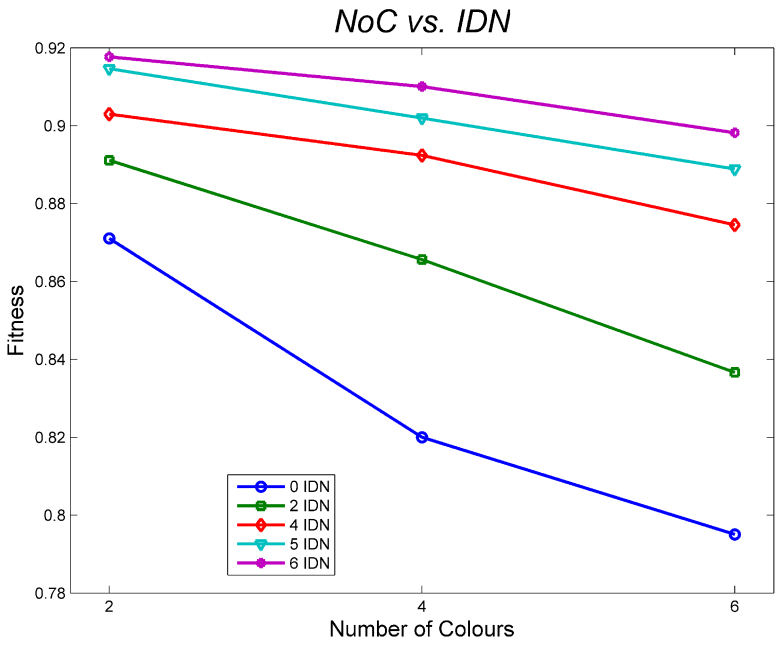


Figure 11.51: Fitness achieved with various number of colours (NoC) and initial number of don't care neighbours (IDN)

Number of Colours vs. Number of Chemical Types

The plot of NoC vs. CHT – see figure 11.52, shows how the effect of changing NoC is dependent on the value of CHT.

In this plot as well, all the lines are nearly linear and shift downwards with increasing CHT. However, in this graph the lines are more parallel compared to NoC vs. IDN – see figure 11.51.

This indicates that changing the value of CHT only has a minor impact on the effect changing NoC has on fitness. Also, it seems like the fitness achieved decreases with increasing CHT regardless of NoC.

The latter is the same effect as was found in chapter 11.3.2. It is interesting to note that the change in impact for NoC on fitness found in chapter 11.3.4 is not seen in this graph. As increasing CHT also makes the search more complicated, the same effect should be expected to appear here. Why it does not is unclear, and more experiments are required to find a good answer to this issue. The data from the experiments performed seems to indicate that IDN and CHT actually play different roles in the development of organisms with various NoC.

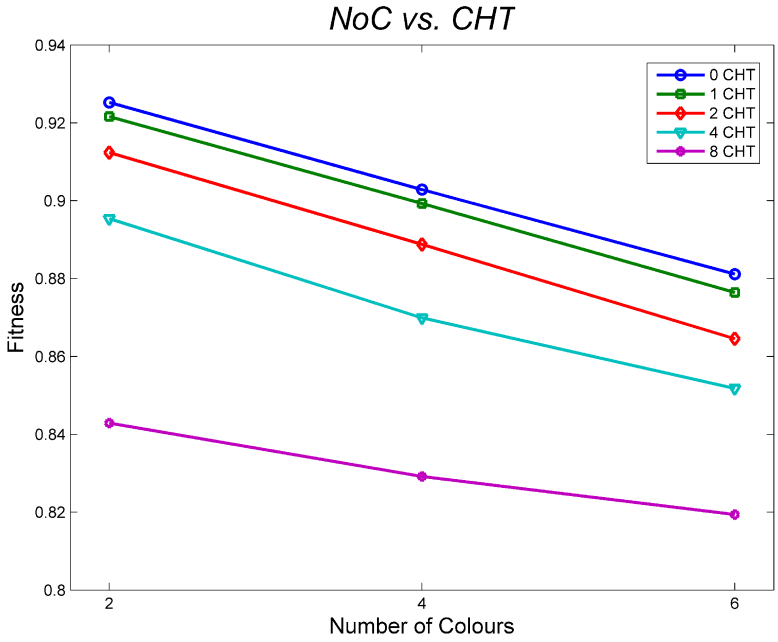


Figure 11.52: Fitness achieved with various number of colours (NoC) and number of chemical types (CHT)

Number of Colours vs. Degrees of Symmetry

The plot of NoC vs. DoS – see figure 11.53, shows how the effect on fitness when changing NoC is dependent on the value of DoS.

Once again, all the lines are close-to linear. As opposed to figure 11.51 and figure 11.52, all the lines here are not running neatly separated; the lines for one and two DoS cross between four and six NoC. For reasons discussed in chapter 11.3.3, zero DoS may be viewed as a special case, and therefore disregarded. So, if disregarding zero DoS, which has the steepest slope of them all, the steepness of the slopes increases with increasing DoS.

As discussed in chapter 11.3.4, further complicating an already complicated search will normally lead to a lower fitness score. However, it was hypothesised in chapter 11.3.3, that as high values of DoS achieve higher fitness than low values of DoS, increasing DoS must make either the search or the development easier. If respecting the hypothesis made in chapter 11.3.4, that is, increasing the complexity of the search will lead to a steeper slope, this means that the search actually has to get more difficult with increasing DoS. This in turn means that the development process must be at least correspondingly more fit to develop shapes with higher DoS to counter this. This is an interesting finding, and a deeper investigation into this issue, requiring new experiments to be performed, should be conducted.

While it might apear significant, the fact that the lines for one and two DoS cross each other is actually not an issue, as this happens because the slope of the line for two DoS is steeper than that of one DoS, and the distance between them is so small.

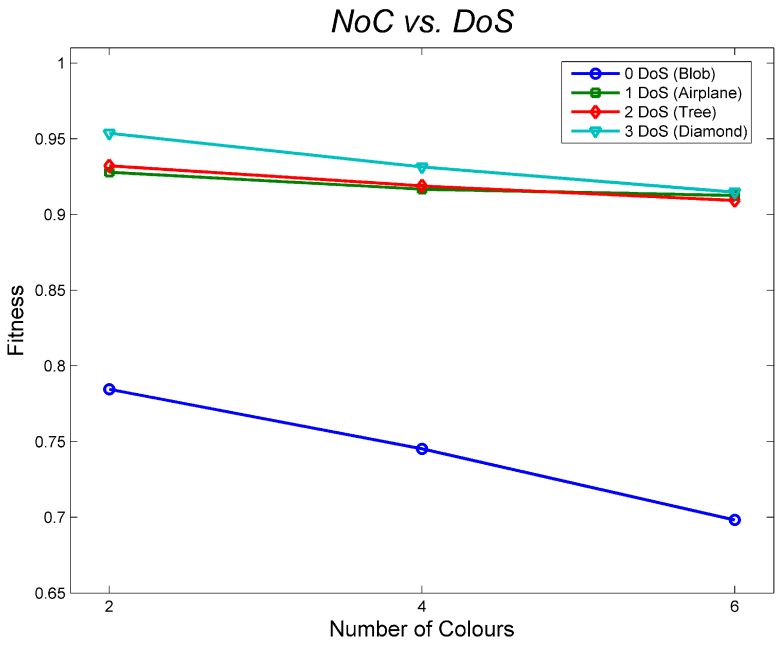


Figure 11.53: Fitness achieved with various number of colours (NoC) and degrees of symmetry (DoS)

Summary

The investigation of how well ArtDev3D is at developing (and evolving) organisms with different number of colours, indicated the existence of a negative correlation between the number of colours and fitness, when disregarding all other parameters. This is as expected, as increasing the number of colours also increases the search space, and hence should result in lower fitness score. Investigating the impact the value of the other three parameters had on NoC's effect on fitness, indicated that the relationship between NoC and fitness was almost independent of their values. This, together with the the fact that all the lines are close-to linear, indicates that the relationship is steady and robust. However, as only three different values of NoC were tested, one should be careful not to overinterpret the meaning of the lines being linear.

11.4 Conclusion

To find the general effect of changing one of the parameters on fitness, the data were plotted as bar graphs with standard deviations. The graphs were then investigated for changes in fitness with changing values for the parameters in question. Both the direction of the change and the regularity of change from one value to the next were taken into account, together with the standard deviations. It was found that both the initial number of don't care neighbours (IDN) and the degree of symmetry (DoS) had a positive correlation with fitness. On the other hand, it was also found that the number of chemical types (CHT) and the number of colours (NoC) had a negative correlation with fitness.

Further investigation, where the values of the other parameters were also taken into account, showed that these findings may have been an oversimplified explanation of the relationship. There exists combinations of parameter values where, for the parameter in question, other values than the expected ones were optimal (with respect to the general correlations found). In particular, there seemed to be great interdependency between IDN and CHT.

Part IV – Conclusion and Future Work

12 Conclusion

A novel artificial development system, ArtDev3D, was presented in this thesis. A number of the mechanisms appearing in biological development were modelled while, at the same time, making sure the system did not become too complex to be useful in practice.

The initial testing of ArtDev3D showed that the system was capable of developing both simple and more advanced three dimensional shapes. The development of simple shapes showed that the system is capable of handle the mechanism of growth, while the more advanced shapes, showed that the system additionally can handle both cell differentiation and morphogenesis.

Further experiments went on to investigate the effect of the system's parameters and their inter-relationship. Through these experiments it was both shown that the system is capable of developing a number of different three dimensional shapes, like a tree or a sphere, and that tweaking the systems parameters may have great influence on the systems ability to develop a given shape.

The final experiments investigated in more detail the effect of changing certain parameters of the system. Four parameters were chosen: the initial number of don't care neighbours, the number of chemical types (both part of the precondition for a protein), the number of available colours (cell types) to the system, and the degree of symmetry of the target phenotype. When concentrating on one parameter at a time, the results indicate that a more stringent precondition (more chemical types, or less initial don't care neighbours) for the proteins makes it difficult for the system to evolve a good solution, at least for the more regularly shaped targets. For the more irregular shapes the results were inconclusive, indicating a more stringent precondition may be required to develop these targets. Also, it was found that an increase in the number of colours available to the system decreases its ability too develop good solutions, with some few exceptions. Finally, the results indicate that using a shape with higher degree of symmetry has a positive effect on the system's ability to develop good solutions. However, when looking at the inter-relationship between the parameters, it was found that this correlation between each of these parameters and fitness was too simple. In particular, there seemed to be great interdependency between the initial number of don't care neighbours and the number of chemical types.

The lack of a set of standard tasks usable for benchmarking development systems is a serious problem in the field of artificial development, making it difficult to compare the various approaches. This issue was already pointed out in [STAN03] and [ROGG04], and suggestions for a suitable set of tasks have been proposed. These suggestions have, however, to date not been established as a standard.

The novel artificial development model proposed in this thesis has through the experiments performed so far shown itself as a potential supplement to other current models. ArtDev3D seems to have a great potential, but further investigation is needed before its real value can be established.

13 Future work

Several issues were not investigated in this thesis because of the time constraint. While it was shown through experiments that ArtDev3D was capable of developing a set of different shapes, and some tweaking with the system's parameters was performed to get a deeper understanding of the system, the systems scalability, effectiveness, robustness and evolvability were not investigated.

The scalability of ArtDev3D can be investigated by using as targets phenotypes having varying size and degree of complexity. A comparison of the relationships between the relative size of the genotype according to its corresponding phenotype for each target will give an idea of the systems scaling behaviour.

The effectiveness of the system can be established through a comparison with other known approaches. This comparison should be performed using target phenotypes having a wide array of different representation sizes. This is advisable because as the development process infers some computational overhead, ArtDev3D is likely to be outperformed when tested with small phenotypes even if it proves itself to be more effective than the alternatives when tested with larger phenotypes.

Robustness towards abrupt changes in the phenotype is a desired feature in an artificial development system, as discussed in 2.3. Achieving this feature was not considered a goal when designing ArtDev3D, but as support for this being an emergent property of artificial development systems has been found [FEDE05b], it is interesting to investigate whether ArtDev3D possesses this feature. Robustness can be tested for by deleting a number of cells in the developing organism at a given stage in the development process and analyse the organism when it reaches maturity for differences compared to a normally developed organism containing the same genome.

Combining an evolutionary algorithm and an artificial development process, introduces problems regarding evolvability. This is because introducing the development process moves the genotype space further away from the solution space, leading to a rougher fitness landscape, and hence a more difficult search. This discrepancy between genotype space and solution space means, for example, that a small change in genotype space may result in a large change in the solution space. Different methods may be applied to increase the evolvability of ArtDev3D, like tweaking the parameters of the genetic algorithm to increase population diversity. Another method, suggested in [FEDE04], is to separate the development process into several distinct stages, where development at each stage is controlled by a different genome.

Bibliography

[BANZ98] Banzhaf, W. Nordin, P. Keller, R. E. and Francone, F. D. (1998). *Genetic Programming - An Introduction*. Morgan Kaufmann Publishers, Inc..

[BENT99] Bentley, P. and Kumar, S. (1999). Three Ways to Grow Designs: A Comparison of Embryogenies for an Evolutionary Design Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), 13-17 July 1999, Orlando, Florida, USA*.

[BONG01] Bongard, J. C. and Pfeifer, R. (2001). Repeated Structure and Dissociation of Genotypic and Phenotypic Complexity in Artificial Ontogeny. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*.

[BONG03] Bongard, J. C. and Pfeifer, R. (2003). Evolving complete agents using artificial ontogeny. In *Morpho-functional Machines: The New Species (Designing Embodied Intelligence)*.

[BROU97] Broughton, T. Tan, A. Coates, P. S. (1997). The use of Genetic programing in Exploring 3D Design Worlds. In *CAAD Futures 97*.

[CAMP96] Campbell, K. H. S. McWhir, J. Ritchie, W. A. and Wilmut, A. (1996). Sheep Cloned by Nuclear Transfer From a Cultured Cell Line. *Nature*.

[CANG94] Cangelosi, A. Nolfi, S. and Parisi, D. (1994). Cell division and migration in a 'genotype' for neural networks. *Network: Computation in Neural Systems*.

[CHEN04] Chen, Y. H. (2004). Simulation of Artificial Development with Focus on Mechanisms for Morphogenesis. Master's thesis, Norwegian University of Science and Technology (NTNU).

[CORB03] Corbalán, L. and Lanzarini, L. (2003). Evolving Neural Arrays - A new mechanism for learning comple action sequences. *CLEI Electron. J.*.

[DELL95] Dellaert, F. (1995). Toward A Biologically Defensible Model Of Development. Master's thesis, Department of Computer Engineering and Science, Case Western Reserve Universe.

[DELL96] Dellaert, F. and Beer, D. B. (1996). A developmental model for the evolution of complete autonomous agents. In *From Animals to Animats IV, 4th International Conference on Simulation of Adaptive Behavior.*

[DITT98] Dittrich, P. and Banzhaf, W. (1998). Self-Evolution in a Constructive Binary String System. *Artificial Life*.

[DUVE95] de Duve, C. (1995). The Beginnings of Life on Earth. *American Scientists*.

[EGGE03] Eggenberger Hotz, P. (2003). Exploring regenerative mechanisms found in flatworms by artificial evolutionary techniques using genetic Parameters in Artificial Evolution. In *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*.

[EGGE04] Eggenberger Hotz, P. (2004). Comparing direct and developmental encoding schemes in artificial evolution: A case study in evolving lens shapes. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*.

[EGGE97] Eggenberger, P. (1997). Evolving morphologies of simulated 3d organisms based on differential expression. In *Proceedings of the 4th European Conf. on Artificial Life (ECAL97)*.

[EGGE99] Eggenberger, P. and Dravid, R. (1999). An Evolutionary Approach to Pattern Formation Mechanisms on Lepidopteran Wings. In *Proceedings of the Congress of Evolutionary Computation*.

[FEDE04] Federici, D. (2004). Evolving a neurocontroller through a process of embryogeny. In *From Animals To Animats 8: SAB 2004*.

[FEDE04b] Federici, D. (2004). Using Embryonic Stages to increase the evolvability of development. In *WORLDS workshop at GECCO04*.

[FEDE05] Federici, D. (2005). Multi-level grounding and self-organization of behaviour through evolution, development, learning and culture. PhD thesis, Norwegian University of Science and Technology (NTNU).

[FEDE05b] Federici, D. and Downing, K. (2005). Evolution and Development of a Multi-Cellular Organism: Scalability, Resilience and Neutral Complexification. *Artificial Life Journal*.

[FOGE66] Fogel, L. J. Owens, A. J. and Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. Wiley.

[GARI94] de Garis, H. (1994). Growing an artificial brain: The genetic programming of million-neural-net-module artificial brains with trillion cell cellular automata machines. In *Proceedings of the Fourth Annual Conference on Evolutionary Programming (EP94)*.

[GOLD89] Goldberg, D. E. Deb, K. and Korb, B. (1989). Messy genetic algorithms: Motiation, analysis, and first results. *Complex Systems*.

[GOLD89b] Goldberg, D. E. (1989). *Genetic Algorithms in search, Optimization and Machine Learning*. Addison Wesley.

[GOLD93] Goldberg, D. E. Deb, K. Kargupta, H. and Harik, G. (1993). Rapid Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms. In *Proc. of the Fifth Int. Conf. on Genetic Algorithms*.

[GORD02] Gordon, T. G. W. and Bentley P. J. (2002). Towards development in evolvable

hardware. In *Proceedings of EH2002, the NASA/DoD Conference on Evolvable Hardware*.

[GRUA92] Gruau, F. (1992). Genetic Synthesis of Boolean Neural Networks with a Cell Rewriting Developmental Process. In *Proceedings of the Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN92)*.

[GRUA94] Gruau, F. (1994). Neural Network Synthesis using Cellular Encoding and the Gentic Algorithm. PhD thesis, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, France.

[GRUA96] Gruau, F. Whitley, D. and Pyeatt, L. (1996). A comparison between cellular encoding and direct encoding for genetic neural networks. In *Proceedings of the First Genetic Programming Conference*.

[HADD01] Haddow, P. C. Tufte, G. and van Remortel, P. (2001). Shrinking the Genotype: L-systems for EHW?. In *Proceedings of ICES 2001, the 4th International Conference on Evolvable Systems*.

[HARI97] Harik, G. R. and Goldberg D. E (1997). Learning Linkage. In *Foundations of Genetic Algorithms 4*.

[HOLL75] Holland, J. (1975). *Adaption in Natural and Artificial Systems*. MIT Press.

[HOLL98] Hollan, J. H. (1998). *Emergence: From Chaos to Order.* Oxford University Press.

[HORN03] Hornby, G. S. Lipson, H. and Pollack, J. B. (2003). Generative Representations for the Automated Design of Modular Physical Robots. *IEEE transactions on Robotics and Automation*.

[JOHA98] Johanson, B. and Poli, R. (1998). GP-Music: An Interactive Genetic Programming System for Music Generation with Automated Fitness Raters. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*.

[JONE93] Jones, A. J. (1993). Genetic Algorithms and Their Applications to the Design of Neural Networks. *Neural Computing and Applications*.

[KITA90] Kitano, H. (1990). Designing Neural Networks Using Genetic Algorithms with Graph Generation System. *Complex Systems*.

[KOZA91] Koza, J. R. and Rice, J. P. (1991). Genetic Generation of Both the Weights and Architecture for a Neural Network. In *International Joint Conference on Neural Networks, IJCNN-91*.

[KOZA92] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Natural Selection*. MIT Press.

[KOZA99] Koza, J. R. Bennett III, F. H. and Stiffelman, O. (1999). Genetic Programming

as a Darwinian Invention Machine. In *Genetic Programming, Proceedings of EuroGP'99*.

[KUMA00] Kumar, S. and Bentley, P. J. (2000). Implicit Evolvability: An Investigation into the Evolvability of an Embryogeny. In *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*.

[KUMA03] Kumar, S. and Bentley, P. J. (2003). *On Growth, Form and Computers*. Elsevier Academic Press.

[KUMA03b] Kumar, S. and Bentley, P. J. (2003). Biologically Inspired Evolutionary Development. In *Evolvable Systems: From Biology to Hardware*.

[KUMA03c] Kumar, S. and Bentley, P. J. (2003). Mechanisms of oriented cell division in computational development. In *Proceedings of the first Aus-tralian Conference on Artificial Life (ACAL 2003)*.

[KUMA04] Kumar, S. (2004). Investigating Computational Models of Development for the Construction of Shape and Form. PhD thesis, University College London.

[KUMA04b] Kumar, S. (2004). Multicellular Development, Self-Organization, and Differentiation. In *GECCO 2004 Workshop Proceedings*.

[KUMA99] Kumar, S. and Bentley, P. J. (1999). The ABCs of Evolutionary Design: Investigating the Evolvability of Embryogenies for Morphogenesis. In *Genetic and Evolutionary Computation Conference (GECCO)*.

[LIU05] Heng Liu and Julian F. Miller and Andy M. Tyrrell (2005). A Biological Development model for the Design of Robust Multiplier. In *Applications of Evolutionary Computing, EvoWorkshops2005: {EvoBIO}, {EvoCOMNET}, {EvoHOT}, {EvoIASP}, {EvoMUSART}, {EvoSTOC}*.

[LUKE96] Luke, S. and Spector, L. (1996). Evolving Graphs and Networks with Edge Encoding: Preliminary Report. In *Late Breaking Papers at the Genetic Programming 1996 Conference Stanford University July 28-31, 1996*.

[MILL03] Miller, J. F. (2003). Evolving Developmental Programs for Adaptation, Morphogenesis, and Self-Repair. In *ECAL*.

[MILL04] Miller, J. F. (2004). Evolving a Self-Repairing, Self-Regulating, French Flag Organism. In *Genetic and Evolutionary Computation -- GECCO-2004, Part I*.

[MJOL88] Mjolsness, E. Sharp, D. H. and Alpert, B. K. (1988). Scaling, Machine Learning, and Genetic Neural Nets. Technical report, Yale University.

[MUEH93] Mühlenbein, H. and Schlierkamp-Voosen, D. (1993). Predictive Models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization. *Evolutionary Computation*.

[PURS03] Purshouse, R. C. and Fleming, P. J. (2003). An Adaptive Divide-and-Conquer Methodology for Evolutionary Multi-criterion Optimisation. In *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*.

[RAVE05] Raven, Johnson, Losos and Singer (2005). *Biology*. McGraw-Hill.

[RECH73] Rechenberg, I. (1973). *Evolutionstrategie: Optimierung Technischer Systeme nac Prinzipien der Biolischen Evolution*. Frommann-Holzboog Verlag.

[ROGG04] Roggen, D. and Federici, D. (2004). Multi-cellular development: is there scalability and robustness to gain?. In *proceedings of Parallel Problem Solving from Nature 8, PPSN 2004*.

[ROOK96] Rooke, S. (1996). The Evolutionary Art of Steven Rooke. *http://www.dakotacom.net/~srooke/index.html*.

[RUBI00] Rubin, G.M. et al (2000). Comparative genomics of the eukaryotes. *Science*.

[SIMS94] Sims, K. (1994). Artificial Evolution for Computer Graphics. *Computer Graphics*.

[SIPP97] Sipper, M. Sanchez, E. Mange, D. Tomassini, M. Perez-Uribe, A. and Stauffer, A. (1997). A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. In *IEEE Transactions on Evolutionary Computation*.

[SPEC95] Spector, L. and Alpern, A. (1995). Induction and Recapitulation of Deep Musical Structure. In *Proceedings of International Joint Conference on Artificial Intelligence, {IJCAI}'95 Workshop on Music and AI*.

[STAN03] Stanley, K. and Miikulainen, R. (2003). A Taxonomy for Artificial Embryogeny. *Artificial Life*.

[THIE99] Thierens, D. (1999). Scalability Problems of Simple Genetic Algorithms. *Journal of Evolutionary Computation*.

[TODD92] Todd, S. and Latham, W. (1992). *Evolutionary Art and Computers*. Academic Press.

[TORR98] Tørresen, J. (1998). A Divide-and-Conquer Approach to Evolvable Hardware. In *Second International Conference on Evolvable Hardware (ICES98)*.

[TUFT03] Tufte, G. and Haddow, P. C. (2003). Building Knowledge into Development Rules for Circuit Design. In *Proceedings of 5th International Conference on Evolvable Systems: From Biology to Hardware, ICES2003*.

[VASS00] Vassilev, V. K. and Miller, J. F. (2000). Scalability Problems of Digital Circuit Evolution: Evolvability and Efficient Designs. In *EH '00: Proceedings of the 2nd NASA/DoD workshop on Evolvable Hardware*.