

# Foreword

After my undergrad studies in Computer Science at the University of Erlangen, Germany, I asked Dr. Jacob of the University of Calgary, Canada, about a possible collaboration in the fields of Artificial Evolution and Artificial Life. A couple of months later, I took an airplane to Calgary and I was introduced to Dr. Jacob's Evolutionary & Swarm Design group. During my visit, I realized that artificial swarms represent far more than exciting models of biological systems. They are models of the scientific paradigm of swarm intelligence: When large numbers of simple units interact, complex emergent patterns can occur. This principle, as seen in natural swarm systems, is also assumed to give rise to our own intelligence and consciousness. Hence, it offers an explanation for important philosophical problems and provides concrete and illustrative examples of many complex natural systems at the same time. These two aspects have nurtured my passion for swarm intelligence.

The present work summarizes my early encounters with the design and evolution of artificial constructive swarms. The original working title of this manuscript, 'Evolving swarms that build 3D structures', has been substituted by the more adequate book title 'Evolving artificial constructive swarms — Experimental models and methodologies'. The new title lends itself to the idea that artificial constructive swarms have become a fast developing discipline in artificial life. As such, it is encouraged to generalize beyond the physical world and to experiment with abstract computational systems.

The presented experiments and their results have coined the investigative course of my studies until today. I hope you will enjoy reading about the beginnings of my research in the exciting area of swarm intelligence.

Sebastian von Mammen  
Paraza, France  
March 2008

# Acknowledgements

I would first like to thank my supervisor Christian Jacob for the generous promotion of my scientific interests. He is a never-ending source of inspiration and motivation. I am very grateful for Gabriella Kókai's comprehensive guidance. She took on the burden to supervise my work at my home university in Erlangen, Germany. This work's implementations could have never been realised without the software and the superb support of Ian Burleigh and Jon Klein. Marcin Pilat and Stefan Mandl have accompanied parts of my work and provided a lot of good advice. Many thanks to all of you. Also, I thank Arne Becker for his tireless efforts to keep my Linux systems running. Sebastian Seifert deserves credit for his support in all questions concerning the Latex environment. Finally, I would like to thank my family for being very supportive and encouraging throughout my studies.

---

## Background and thesis objective

**Topic:** Evolving swarms that build 3D structures

**Background:** Swarm intelligence systems are attracting more and more attention recently, as many design decisions nowadays have to incorporate dynamical systems with a very large number of interacting agents. One example is the design of city and, in particular, traffic infrastructures. Questions of where to build new roads, bridges, tunnels, intersections, etc is largely dependant on simulations of traffic patterns, as obtained through the monitoring of car and pedestrian traffic. Simulations, in which emergent traffic patterns are analyzed, have to be performed to evaluate different options for improved or additional infrastructure components. Hence, this iterative (and in parts evolutionary) design process involves collective intelligence systems (such as car agents or pedestrian agents). The Evolutionary & Swarm Design group at the Department of Computer Science, University of Calgary, is investigating simulations of emergent patterns in swarm intelligence systems. We are especially interested in models of social insects (army ants, termites), bacterial ecosystems, and of biomolecular systems (such as protein-protein interactions within a cell).

**Thesis objective:** For the context of this student thesis we will focus on simulations of social insect behaviours. In particular, we want to investigate building behaviours of social insects, such as the construction of termites' nests. In [1] and [2] an evolutionary system is described, which programs swarm behaviours through interactive 'breeding'. The implemented system uses an evolutionary kernel (Evolvica [3, 4]) and an agent-based swarm simulator (Breve [5]). The thesis objective is to build on these two systems to implement an evolutionary design system for swarms, that model social animals (such as insects or birds) and their building behaviours (such as termites building their nests). The approach should, however, not focus on how a particular swarm system performs its building tasks, but rather explore an evolutionary way for getting the swarms to construct 3-dimensional structures. This could, for example, be achieved by having the swarm agents move small, stackable objects in a 3D environment. The actual control programs will be not manually designed but by interactive and/or automated breeding, following the well-tested approach in [2].

---

**Literature:**

1. Kwong, H. and C. Jacob. Evolutionary Exploration of Dynamic Swarm Behaviour In Congress on Evolutionary Computation, 2003, Canberra, Australia: IEEE Press.
2. Kwong, H. Evolutionary Design of Implicit Surfaces and Swarm Dynamics Master Thesis, University of Calgary, May 2003.
3. Jacob, C. Illustrating Evolutionary Computation with Mathematica Morgan Kaufmann Publishers, San Francisco, 2001.
4. IEC web site. <http://pages.cpsc.ucalgary.ca/~jacob/IEC>
5. Jon Klein. <http://www.spiderland.org/breve>

**Supervision:** Christian Jacob and Gabriella Kókai

# Abstract

This thesis presents two different approaches to evolve swarms that create three-dimensional structures. Consequently, both approaches comprise a swarm model serving the constructional purpose and the design of an adequate evolutionary system.

Basis of an appropriate swarm agent are an existing model of a swarm's flight [30] and the knowledge that certain insects' construction abilities result from inherited behaviour patterns [12]. In detail the agent's sensory abilities, its inference system and its set of actuators are specified. According to the knowledge about insects' coordinated work, communication happens indirectly through manipulation of and reaction on the environment. The presented model determines the agent's general behavioural capabilities but does not specify its actions for a given situation.

In fact, the agent's flight and construction behaviour can be understood as its *genotype*. It has to be evolved to accomplish the given task to create three-dimensional structures. Whenever creative or interesting objectives had to be evolved, *interactive evaluation* has stood the test: An external supervisor guides the search by influencing the outcome's (also *phenotype*) *fitness* rating.

In the first approach, also called the *connectionist* approach, the individuals' decisions result from processing the weighted sums over all available information. That is information describing the swarm individual's situation as well as its environment. The weights of this "behaviour network" are *mutated* and exchanged among the swarms (*crossover*) to please the supervisor. This approach has failed because the computation of a single *generation's* set of phenotypes takes too long for interactive evaluation. Nevertheless, it may be stated that a connectionist swarm model offers a way to create smooth, colourful and diverse structures.

The individuals of the second approach, also referred to as the *rule-based* approach, base their creative decisions only on the structure that surrounds them. This behaviour is

---

encoded in a set of conjunctive rules. The weights that determine the flocking behaviour are evolved together with these rules. A predefined three-dimensional shape guides the evolutionary search. The successful approximation of this given 3D object is rewarded with a positive fitness value. Several rule-based swarms have been found that build interesting structures. Additionally, a way to interactively guide the search by using alterations of the approximated structures has been successfully applied.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Seminal models and ideas</b>	<b>5</b>
2.1	Techniques and examples of nest building in social insects .....	5
2.2	Flocking .....	10
2.3	Creative and aesthetic design through interactive evolution .....	12
<b>3</b>	<b>Approaching a constructive swarm</b>	<b>18</b>
3.1	Modeling a swarm that builds three-dimensional structures .....	18
3.1.1	Desirable results .....	19
3.1.2	The relation between a swarm and its individuals .....	19
3.1.3	The individuals' basic abilities .....	20
3.1.4	Perception, data processing and action of the swarm individuals....	21
3.2	Connectionist approach .....	23
3.2.1	The individuals' construction operators .....	23
3.2.2	Behaviour determined by weighted input .....	24
3.2.3	Interactive evolution .....	29
3.2.4	Genetic operators .....	29
3.2.5	Implementation details .....	32
3.3	Rule-based approach .....	34
3.3.1	The individuals' operators .....	36
3.3.2	Behaviour determined by "if-then-rules" .....	39
3.3.3	Approximation of a given 3D structure .....	40
3.3.4	Genetic operators .....	41

3.3.5 Implementation details.....	43
<b>4 Evaluation of the two approaches</b>	<b>46</b>
4.1 Strengths and weaknesses of the connectionist approach .....	46
4.1.1 An example: Construction of maximum height .....	46
4.1.2 Fatal inefficiency.....	48
4.1.3 Nice features of the connectionist approach .....	50
4.2 Evaluation of the rule-based approach .....	50
4.2.1 Exemplary structures .....	51
4.2.2 A case-study: Approximation of a tower .....	56
4.2.3 Guiding the search with diversified 3D objects .....	58
<b>5 Summary and future work</b>	<b>64</b>
<b>A Appendix, DNA of a Tower Building Swarm</b>	<b>66</b>



# List of Figures

2.1	Pilat's lattice-swarm has successfully built a construction resembling the nest of the wasp family <i>Agelaia</i> . . . . .	8
2.2	The image shows the step-wise course of construction. If a microrule algorithm is coordinated, asynchronous work does hardly affect the outcome - the sequence of the single steps would play no role. If the building rules lead to repeating patterns, a structure is created fairly easily. . . . .	9
2.3	Interactive evolution led the swarms to new architectures ("Plateaux" and "Tunnels" by Pilat [29]). . . . .	11
2.4	The picture illustrates the three different urges of Craig Reynold's original swarm model: Cohesion, alignment and separation. The resulting acceleration vectors are coloured in red. Neighbourly flockmates in blue. . . . .	12
2.5	The first two images show the same swarm. Due to its tendency to switch from rings to sinuous lines, Jacob and Kwong named it "Big Ring Snake". On the right is a typical eight formation [23]. . . . .	16
3.1	These are sample images of Jacob's "Towers" simulation. The colourful arrows are the swarm individuals. As soon as they reach the ground they start piling up spheres. . . . .	25
3.2	The "behavioural net" of a swarmette: The left side shows the available input data. In the middle part there are the computational units - perceptrons are denoted with a "p". The output on the right, which depends only on the weights along the edges, is interpreted as the individual's set of actions. Most terms represent vectors, except for the green ones. Each computational unit "fires" one scalar. . . . .	28

3.3	The course of interactive evolution implemented to evolve a “connectionist” swarm. . . . .	30
3.4	The update $\Delta v$ that is added or subtracted from the original weight value depends on a random value $x \in [0, 1]$ . It can be seen that minor changes have higher probability. . . . .	32
3.5	An agent, represented by a yellow sphere, is going to collide with the green cubic construction particle (the agent’s velocity is indicated by the blue arrow). The agent’s behaviour depends on the structural configuration that surrounds the collision partner. In the illustrated case red vectors point to the locations that will be checked for particles by the agent. . .	35
3.6	To analyze the surrounding structural configuration, the agent tests, whether there are bricks at certain positions or not. The figure illustrates the following case: An agent $s$ collides with a particle $c$ . One of the rules of $s$ checks for a brick at $\vec{p}_i$ relative to the position $\vec{p}_c$ of the collision particle. . . . .	40
4.1	The construction phenotype of a randomly initialized connectionist swarm population. . . . .	47
4.2	Fourteen generations later, the constructions of Figure 4.1 have evolved to this set of structures. The higher a structure was, the better was the swarm’s rating. . . . .	48
4.3	The location vector of the agent strongly influences the perceptron’s decision to build particles. The number of built particles varies according to the agents’ position. . . . .	50
4.4	On the left there is a line of seed blocks that trigger the swarm’s building process. The right image shows the swarm’s constructional result. The almost equal distribution of the swarmettes along with their low flight supports their constructional efforts. . . . .	53
4.5	The fan-like construction has been achieved by starting from a single seed block in the simulation world’s center, a small distance from the ground. The image on the left side shows an earlier stage of the construction’s development. . . . .	53

4.6	The swarm is divided into two flocks at an early stage. Both flocks loop back and forth from their sides to the construction. The many holes in the building make it look like a bush. . . . .	54
4.7	Two-level flats: Developed from two seed blocks at the corresponding heights. The third image shows the construction from above. . . . .	54
4.8	The more interesting shapes are unsymmetrical like this one. The right picture enhances the view on the swarm’s flocking behaviour. . . . .	55
4.9	An excellent example of an asymmetric, organic looking shape. But it has only two dimensions: The right picture shows the structure from the side, no swarmette ever leaves the plane. . . . .	56
4.10	Intermediate states of the building process of a tower. The orange “fitness structure” elucidates the excellence of the swarm’s approximation. . . . .	60
4.11	The swarm’s flocking behaviour after it has successfully approximated the given tower structure. If the swarm gets into contact with the central building it might be triggered to extend the construction. Since particles outside the given structure is punished, the swarm is better off hiding in the world’s corners. . . . .	61
4.12	Extended versions of the original guiding structure (seen in the last image of Table 4.1): An extension of the tower’s height, a branching “crown” and stairs on top of the tower. . . . .	62
4.13	1. Pillars arise after 1000 generations 2. Extension of the original “fitness structure” results in endless efforts to gain height (644. generation) Both images clearly show the vertical line flight formation that contributes to the swarm’s construction . . . . .	62
4.14	After 1341 generations the original tower building swarm has adjusted to the new challenge: Now it contributes to the branching structure by extending its construction’s diameter with growing height. . . . .	63
4.15	The first image shows the result of an evolutionary run with an optimized tower building swarm as basis. On the right side an outcome without a specific initialization is displayed (925. generation). . . . .	63

# List of Tables

3.1	Contrasting the general differences of the two approaches . . . . .	22
3.2	Set of actions of a connectionist agent . . . . .	24
3.3	The connectionist agent's available information (16 floating point and 13 integer values) . . . . .	26
3.4	These values determine the connectionist agent's swarming behaviour (after [35]). . . . .	27
3.5	Default parameters of the connectionist approach's implementation . . .	33
3.6	The rule-based agents' actuators . . . . .	38
3.7	Settings of the rule-based approach's phenotype simulation and evolution process . . . . .	45
4.1	Structures that have guided the evolutionary search are coloured in red, seed blocks in blue. The corresponding results are referenced on the right side. Furthermore it is stated in which generation $g$ the result has occurred and what fitness $f$ it has achieved. . . . .	52
A.1	The rule-based agent's actuators corresponding to the genotype numbers.	68

# List of Algorithms

2.1	Simple Genetic Algorithm, by Mitchell [27]	17
3.1	Volume of Two Cubes' Intersection	42

# 1 Introduction

Cars stuck in a traffic jam on the highway, fish that swim in schools, wasps that create nests, cells forming a human organ or people living in big cities - all of them can be seen as units of an *emergent* system. Emergence means that the *complex* result has qualities that are not present as long as the single units are isolated from each other [9]. Whereas “complex” indicates that the connections and dependencies of the units’ interplay have had a major effect on the outcome [31].

Seeking solutions to complex tasks becomes more and more important. Whether it’s the jurisdiction of a democratic state or the blueprint of an automobile - complex systems tend to grow, since “it’s always easier to add components to such a system than to remove them” [33]. In designing software systems this happens to even “threatening” extents [12]. Fortunately, scientists have found a way to tackle this problem. Instead of mastering a complex task by engineering every single detail, one tries to figure out how the whole system can be understood as an incorporation of many simple, *autonomous* functional units (called *agents*). The interactions taking place in such a *multi-agent system* are not predefined or controlled by a central instance, but coordinated by the different agents themselves [8], in other words they are *self-organised* [12].

The elegance of this approach lies in the agents’ actions and interactions. However, discovering these mechanisms can be a very difficult task [12].

Once again scientists can fall back upon the means of conservative engineering as most groups do in the popular *RoboCup*-competition [2]. Here the task is to come up with a perfectly incorporated group of autonomous robots which plays a variant of the soccer game.

If a well-defined task is given, another approach can be the search for the agents’ optimal behaviour. By means of a genetical algorithm, Mandl has found effective be-

havioural rules for the protagonists of the Predator-Prey Pursuit Game<sup>1</sup>. The prey is captured (which means the prey is surrounded by the predators), as the predators communicate and work together [25].

In the work “The Society of Mind” Minsky postulates that even intelligence might emerge from the incorporation of a set of different agents [26]. Though the method of designing this emergent intelligence - namely reasoning - may be controversial, the general scheme he has drawn is still convincing.

Nevertheless, reasoning has already brought some other emergent systems to light. Playing with the possibilities of cellular automata made clear that iterative application of rules which form the environment can produce emergent patterns (e.g. “The Game of Life” [4] or “Langton’s Ant” [38]).

The most direct way to find mechanisms that yield the solutions to complex problems, is the detailed observation of natural emergent systems. Modern biology makes it possible to understand the mechanisms within a living cell as interactions of amino-acids and other molecules. With this agent-based point of view, scientists have already made great steps in simulating parts of this immensely complex system [13].

Insect swarms are a representative objective of studies about emergent occurrences in nature. They consist of many autonomous beings. Each of them acts on its own and in total they form an emergent whole. The movement of a swarm might be its most obvious characteristic. But these swarm individuals do not only coordinate their acceleration and velocity. Together they accomplish complex tasks like organising resources or building nests [12]. Implementations of a simple *rule-based lattice-swarm* affirm the current model of such insects’ coordination (more in Chapter 2), which was the starting point that inspired us to this thesis’ objective: Evolving swarms that build three-dimensional structures.

In other words, it is the search for evolutionary ways to develop a swarm that creates structures in 3-dimensional space. Except for the mentioned model of creative insect swarms, there has not been another similar approach. Additional functionality of the swarm individuals brings more possibilities in the creation of 3D structures. In contrast to a mere implementation of flocking agents that act upon predefined patterns, this

---

<sup>1</sup>In this case searching for the agents’ emergent behavioural rules may be obvious, as the game itself suggests the existence of a set of roles/agents.

work tries to figure out how such agents could be bred in an evolutionary process. Consequently, this thesis investigates the two following major subjects:

1. The general design of the swarm individuals, concerning their sensory abilities, information processing system and actuators.
2. An evolutionary framework to search for swarm behaviour that succeeds in 3-dimensional construction.

With a certain degree of randomness the considered swarm individuals are determined in their behaviour. The questions that have to be answered by a detailed implementation are: *What kind of knowledge can these agents possibly perceive?*, *How is their input processed?* and *Which actions is an agent able to execute?*. The next step is to make up one's mind about the internal representation of the agent's behaviour. It must be suitable for an evolutionary algorithm and it should neither broaden nor limit the search for an appropriate swarm too much. If thousands of values determine the agent's behaviour, the search for a good combination of values might be inefficient to compute. On the other hand, if the agents have too few information about their environment, they might not be able to make the necessary decisions to create interesting structures (the corresponding question would be: *How complex does an agent have to be in order to create structures with its mates?*). Of course, the successful search depends heavily on the genetical operators' functionality. *Crossover* and *mutation* generate new swarms, *selection* guarantees development into the desired direction. The guidance of evolution plays an especially important role. As there is no mathematical concept of "beautiful" or "interesting" structures, and therefore no direct way to automatically evaluate the swarms' constructions.

Two separate approaches to evolve a "construction swarm" are presented in this work. The first one, referenced as the *connectionist approach*, implements the individuals' information processing system as an artificial neural net. A great number of input values is provided and the agents' actuators leave a lot of freedom, too. The evolutionary process is supposed to be guided by a human supervisor, interactively rating the swarms' products. Although the simulations' results looked "nice" right from the start, the time needed to adjust the individuals' behaviour was not acceptable. The interplay of a huge search space (there were more than 300 floating point numbers to adjust) with time



needed by the simulation runs (computation and visualization took about three minutes for the results of one generation) slowed down the process too much. That is why the next approach follows a different leading idea. In the second approach, also referred to as the *rule-based approach*, the agents' behaviour is determined by a set of rules that considers only the individual's local environment. Furthermore, the evolutionary process happens automatically. A predefined three-dimensional object guides the search. Swarms that align their construction with this object have a greater fitness, and, thus, higher probability to be transferred into the next generation. Simulations have shown that this approach yields swarms that orientate themselves in the given structure but also develop their own creativity.

Chapter 2 starts with presenting some examples of how natural "swarm intelligences" construct their nests. Furthermore, an appropriate model for the agents' movement in space is introduced together with some recent discoveries in this field. After explaining basic evolutionary techniques, some cases of human-guided evolution round off the chapter. A discussion about the general abilities of a "construction swarm's" individual introduces the third chapter, which is then followed by detailed explanations of the connectionist and the rule-based approach. In Chapter 4 the insights provided by the experiments of both approaches are presented. Although this work cannot cover all possibilities in designing a system to evolve a structure building swarm, a huge spectrum of choices has been looked upon. After a short summary of this thesis, an outlook on a possible consequent approach is given in Chapter 5.

## 2 Seminal models and ideas

“Evolving swarms that build 3-dimensional structures” is a very extensive task. *Biology* gave birth to its underlying ideas. A model to simulate a swarm’s flight has been successfully discovered and implemented by the scientist Craig Reynolds working in the *computer graphics* sector. Evolutionary algorithms are a popular optimization method in *various disciplines of computer science*. The direction of this evolutionary search is heading towards a goal somewhere between the emergence of “creativity”, “aesthetics” and the coordination of autonomous agents, including possible applications in fields such as *self-assembly, robotics, architecture* and *fine arts*.

This chapter leads through the maze of distinct scientific branches to provide the information needed to understand the given task.

### 2.1 Techniques and examples of nest building in social insects

The inspiration for this thesis’ objective are swarms of social insects that build nests. Many of these swarms have been object of studies themselves and in the book “Swarm Intelligence - From Natural to Artificial Systems” [12] some of their occurrences and ways to explain them are summarized. The authors Bonabeau, Dorigo and Theraulaz have placed their main interest in analysing the mechanisms of individual and cooperative actions that result in emergent accomplishments. The gained knowledge can be used to master similarly challenging tasks in computer science and other fields of engineering.

In opposition to an *anthropomorphic* approach to building nests, which basically means that each individual follows a blueprint of an architecture, scientists nowadays claim that an insect’s behaviour depends on local information and is determined by a “*simple probabilistic stimulus-response*” scheme. Communication between the swarm individuals

happens as an important by-product of this behaviour - one individual alters the environment and another one reacts to these changes. This kind of indirect communication is called *stigmergy*. Two cases are differentiated:

1. *Quantitative stigmergy*, where triggering a reaction depends on the intensity of a stimulus.
2. *Qualitative stigmergy*, where an individual reacts on a discrete occurrence of some sort.

It may be assumed that the latter kind plays the dominant role in building complex nests. For example, discrete stimuli can be the particles that have already been built or the environmental structures that are provided by nature. Any occurrence that initiates and guides a certain building behaviour is called *template*.

The number of different stimuli goes hand in hand with the number of particles built. So, while a construction grows in size, the building rate per individual increases, too. This phenomenon of *self-organisation* and a large number of individuals make it possible that some social species exceed solitary ones by far in their nests' complexities.

Bonabeau et al.'s first example of insects' building behaviour presupposes the existence of a template and a self-organisation mechanism. In this case the template is the body of a termite (*Macrotermes subhyalinus*) colony's queen that spreads a pheromone trace. The worker termites start building protecting walls when encountering the right level of the queen's pheromone. As soon as the first pellets (construction elements that diffuse another kind of pheromone, called *cement pheromone*) are deposited, a *snow-ball effect* crops up. Single workers that are in the near distance to these pellets are attracted and contribute to the building process. The resulting building is adapted to the queen's body. Based on Deneubourg's earlier studies [7], Bonabeau et al. show how a *reaction-diffusion* model of this *construction of the royal chamber* can be designed.

Similarly, the ant *Leptothorax albipennis* builds its nest's walls. The brood together with some ants forms a circular cluster. And though it is unclear, whether this template functions on chemicals or the ants perceive its physical presence, workers start constructing walls in a certain distance. Once again, this process accelerates after a minimal amount of building elements has been deposited. Formalising the probability of

construction depending on the proper distance and the perceived density of construction material leads to a good simulation of the natural occurrence.

With these two examples, termites that construct the royal chamber and ants that build the walls of their brood-nest, Bonabeau et al. demonstrate the successful interplay of templates and self-organisation. In contrast to these cases, in which the insects mainly act in accordance with quantitative stigmergic stimuli (e. g., pheromone concentration that passes a given threshold), the eusocial wasp *Polistes dominulus* builds determined by qualitative heterogeneities. Constructing a wasp-nest starts with a so-called *pedicel*, onto which workers attach rows of cells (all construction elements consist of carton which is made of plant fibres). Before starting a new row, a worker tends to fill existing gaps. This shows that the local configuration determines the wasp's action. Furthermore, this coordinated behaviour allows to work in parallel without disorganising the construction process.

Bonabeau et al. have designed an agent-based model to examine the nature of the individuals' cooperation (in a discrete three-dimensional world). In this model an agent deposits a *brick* (one of two kinds) at the current position, if the conditions of one of its *microrules* are satisfied. Whereas, a microrule is a concatenation of conditions that test the situation of the agent's neighbourhood at fix locations. Two ore more microrules that are triggered in the same situation are considered *incompatible*, because the agent is supposed to deposit only one brick at a time. A collection of microrules can then be seen as an *algorithm*. There are several phases of creating a nest. To ensure that a nest can be built by such an algorithm, it is crucial that local configurations and the necessary constructions in each stage do not conflict with each other. If this requirement is met, the algorithm is *coordinated* and the wasps can work in a parallel fashion.

Choosing a microrule algorithm randomly does not yield interesting structures. However, designing a microrule-set backwards, starting with the shape of the wasps' nest, leads to success. Using this approach, constructions similar to those of the wasps *Epipona*, *Parabolybia*, *Stelopolybia*, *Vespa* and of the *Chatergus* genera were computed by Bonabeau et al. Pilat was not only able to reproduce these results, but also to come up with rule-sets for the wasp families *Agelaia* (Figure 2.1), *Parachatergus* and *Vespula* (see [29]).

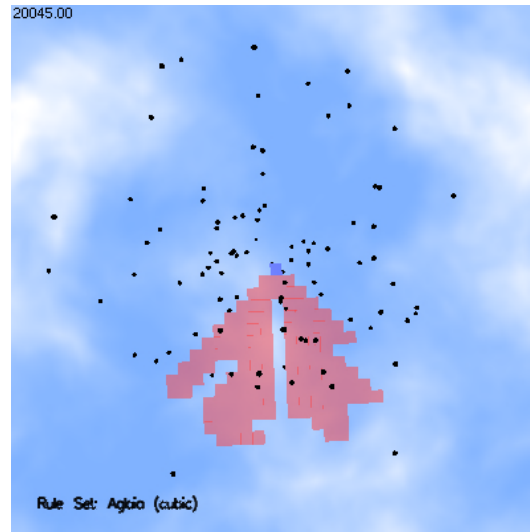


Figure 2.1: Pilat’s lattice-swarm has successfully built a construction resembling the nest of the wasp family *Agelaia*.

The set of microrules determines the nest’s general shape. The output is not always totally determined. As the agents act asynchronously, minor differences in the course of construction can strongly affect the outcome. Nevertheless, certain attributes of the set of microrules can be mapped onto the resulting shapes.

There are three common descriptions of *structures* [10]:

1. The way in which something is organized, built or put together.
2. A particular system, pattern, procedure or institution.
3. A thing made of several parts put together in a particular way.

It is assumed that structures only result from coordinated algorithms. Without coordination, the parallel work of insects would result in disorganisation of the building behaviour and therefore in an unstructured shape. If patterns (or *modules*) are repeated by a coordinated algorithm, a structured output is highly probable (see Figure 2.2). During the experiments it has become clear, that the number of microrules used by an algorithm is usually linked to the arising structure’s complexity - a few or even only one applied microrule in the algorithmic process indicates non or only a poor structure.

From the building element’s viewpoint the described model realises a form of self-assembly. This analogy becomes more obvious when looking at the general definition of

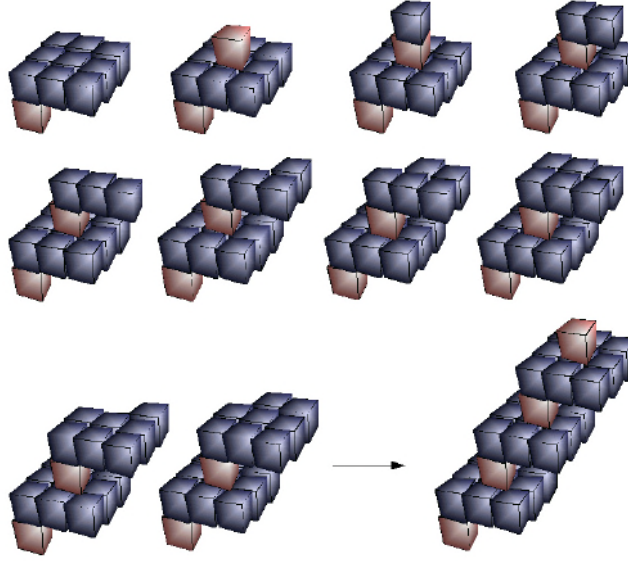


Figure 2.2: The image shows the step-wise course of construction. If a microrule algorithm is coordinated, asynchronous work does hardly affect the outcome - the sequence of the single steps would play no role. If the building rules lead to repeating patterns, a structure is created fairly easily.

an assembly task [20]. The task of a very basic assembly system brings an initial assembly  $E_0$  to a final assembly that is element of a set of possible solutions,  $E_N \in \mathcal{E}_{final}$ . In the lattice-swarm construction algorithms the given pedicel/seed is equivalent to  $E_0$ , and  $E_N$  is the resulting nest or construction. During the single steps of the assembly process, configurations might appear that would lead to failure (summed up in set  $\mathcal{B}$ ). Excluding any of these “obstacles”, which is desirable in self-assembly systems, is very similar to the informal definition of structure creating, coordinated algorithms. No intermediate situation should lead to an impasse:  $E_t \notin \mathcal{B}$  with  $t \in \{0, \dots, N\}$ . Of course, the coordination of an algorithm influences the shape of the nest. On the other hand, one can say that only a subset of nest-shapes is suitable for in parallel working social insects.

Furthermore, coordinated algorithms are supposed to be easier mapped onto the resulting constructions. This can be derived from the fact that non coordinated algorithms have a great tendency to deviate from preceeding outcomes in each run.

Analysing the results mathematically suggests that there is only a small group of (very similar) algorithms that leads to *compact* constructions. One speaks of compactness if many bricks are connected with each other. This “requires collections of complementary, correlated microrules.”

Manually designed microrule-sets represent only a small part of the vast amount of possible algorithms. To guide the search through the space of architectures, Bonabeau et al. use heuristics from which conclusions about the structure can be drawn (as mentioned above) and they also integrate the evaluation of external supervisors.

In this way many architectural styles far from wasp nests have been discovered. Pilat evolved some more patterns and already started describing and classifying them (see Figure 2.3). In addition, he could show experimentally that a proper architecture depends on a minimal number of agents. During his experiments he realised that the world’s fix orientation did not allow some rule-sets to be applied to their full potential. Though a local configuration was at hand that should have triggered an agent’s rule, it failed as the microrule’s conditions were tested in respect to only one general direction. The suggested solution to this *symmetry problem* is testing the agent’s rules according to its own orientation. Finally Pilat demonstrates that minor changes in some rules can have major effects on the constructions built.

## 2.2 Flocking

In the previous section some basic building techniques of insect swarms and colonies have been presented. In the mentioned algorithmic models the individuals’ movements happen randomly. However, talking about swarms implicitly means a certain spatial coordination of the individuals. In the late eighties Craig Reynolds suggested a model for the complex phenomenon of swarm flocking [30]. This model originates from several urges (*steering behaviours*) by which each bird seems to be driven. Think of a flock of birds. No bird wants to crash into one of its flockmates - so it keeps a safe distance to all its neighbours. This urge works against crowding and is therefore called *separation*. In order to keep up with the rest of the flock, each bird has to adjust its heading in accordance with the average direction. This *alignment* takes care of a uniform motion. The third necessary behaviour is that each bird tends to the center of its local flockmates

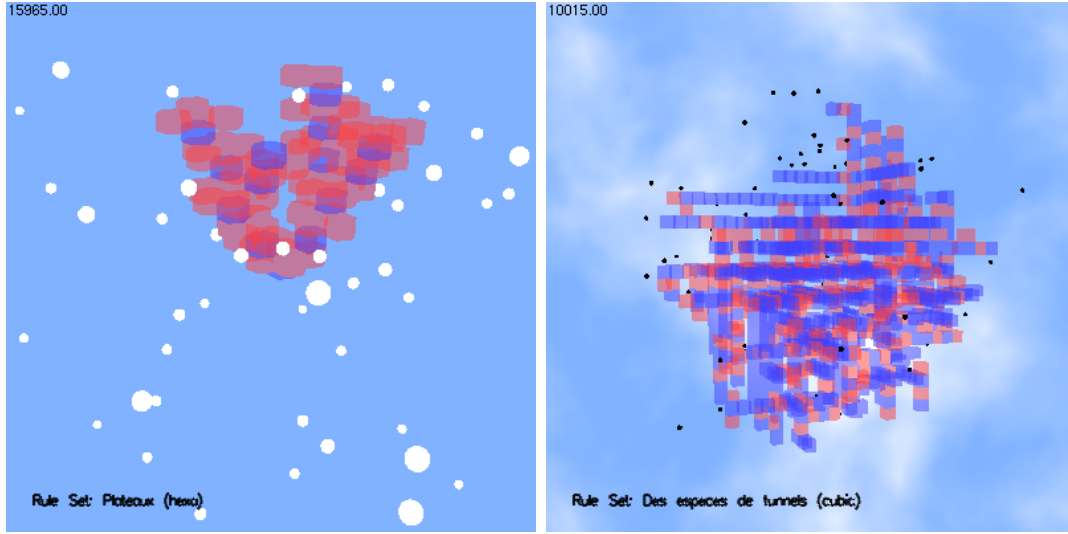


Figure 2.3: Interactive evolution led the swarms to new architectures (“Plateaux” and “Tunnels” by Pilat [29]).

(*cohesion*), otherwise a flock could not be held together. Furthermore, an individual’s perception is limited - it reacts only to mates in its *neighbourhood*. Whereas the neighbourhood encloses all other individuals that are within a predefined range and angle (illustrated in Figure 2.4).

From time to time, new factors or variants of the preceeding ones are proposed to gain a specific swarming behaviour. One possible extension is the individual’s preference for a “better view” - the resulting flock shows the typical “V formation” of migrating birds [15].

Weighting the basic urges to different extents opens up a huge space of formations and flocking behaviours. Jacob and Kwong have discovered parameter sets to simulate distinct flight patterns (a. o. flocking behaviour, line formations, figure-eight and ring formations, see Figure 2.5) [23]. To compute an individual’s acceleration in each simulation step, they considered the weighted sum of the following vectors (after [35]):

$\vec{V}_1$  the opposite direction of all neighbours within distance  $d$ .

$\vec{V}_2$  from the agent towards the world’s center.

$\vec{V}_3$  the average of the neighbours’ velocities.



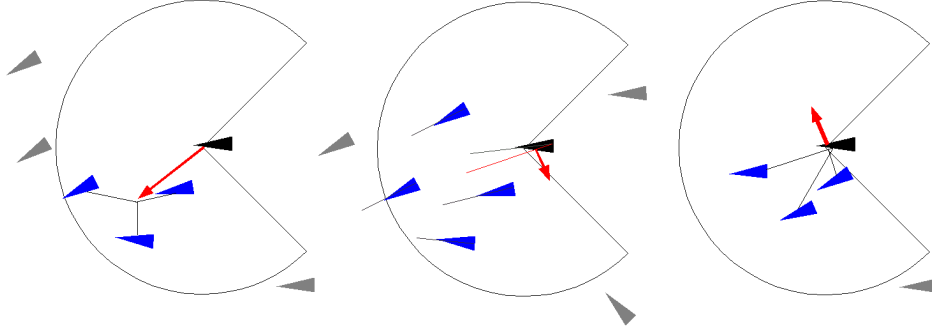


Figure 2.4: The picture illustrates the three different urges of Craig Reynold's original swarm model: Cohesion, alignment and separation. The resulting acceleration vectors are coloured in red. Neighbourly flockmates in blue.

$\vec{V}_4$  the center of gravity of the agent's neighbours.

$\vec{V}_5$  a random unit-length vector.

The mentioned set of parameters includes the scalar coefficients ( $c_1 \dots c_5$ ) of these vectors, the *crowding radius*  $d$  and the maximal acceleration and velocity ( $A_{max}$ ,  $V_{max}$ ). Each parameter influences the swarm's flight. And the consequences of changes of the values can hardly be foreseen. Jacob and Kwong were able to find interesting flight formations in an *evolutionary* process guided by *interactive* evaluation of different parameter settings. In the next section this approach is presented in detail.

## 2.3 Creative and aesthetic design through interactive evolution

The third source of inspiration for this thesis are the many promising results of interactive evolution. First of all the idea of a *genetic algorithm* is explained [28], [27]. After the discussion of its general attributes, the extension of *interactive evaluation* (also *interactive evolution*) is presented. Some examples of this method's successful application round off this chapter.

A genetic algorithm basically implements a *generate-and-test beam-search* through the space of all possible hypotheses (*hypothesis-space*). This is accomplished by keeping and updating a set of solutions (in evolutionary/biological terms: A *population* of *individuals*) in accordance with the quality (*fitness*) of each hypothesis' output (*genotype* and *phenotype*). The population's offspring are generated and tested by so-called *genetic operators*. The standard set of these operators consists of:

**Selection** is responsible for choosing the offspring of the current population according to the individuals' fitnesses.

**Mutation** randomly changes parts of an individual's genotype.

**Recombination/Crossover** generates a new individual by combining parts of its ancestors' genotypes.

The so-called *simple genetic algorithm (SGA)* [28], which implements the most common variations of selection, mutation and crossover, is shown in Algorithm 2.1.

Until now it has not become totally clear, how the efficient computation of proper results emerges from the interplay of these genetical operators. However, it has been proven that under certain circumstances an *implicit parallelism* (this term was introduced by John Holland in 1975 [17] and has recently been revised by Wright et al. [14]) speeds up the search through population-space.

Well-known facts about GAs are that they are quite robust in avoiding local optima and that they are highly adaptive. The latter attribute refers to the ability of adjusting the populations' predominant genome to a recently changed environment (or fitness-rating) in only a few generations [18] and [19].

Of course, for each of the mentioned genetic operators there exists a great variety of extensions and derivatives<sup>1</sup>. In addition, more operators as well as some methods to increase the power and/or speed of the evolutionary process have been suggested. Useful concepts for this thesis' implementations were:

**Elitism** takes care of the best solutions' unquestioned propagation to the next population [6] (extend the selection of the SGA 2.1, line 5, with: Take  $e$  percent of the members

---

<sup>1</sup>For a more detailed introduction in evolutionary algorithms see for example [27]

of  $P$  that have highest fitness and add them to  $P_s$ . Continue proportionate selection with only  $(1 - p - e)p$  members).

**Multi-start hillclimber** ensures that the good solutions found can be used as a starting point for further investigation [28] (change the first line of the SGA 2.1 to: *Initialize population:  $P \leftarrow p$  hypotheses that have already been computed and have achieved promising results*).

**Interactive Evaluation** enables an external supervisor to rate the phenotypes' fitnesses or at least to partially influence their fitness values [11] (implement the function  $Fitness(h)$  used in Algorithm 2.1 in such a way that a supervisor can rate the hypotheses).

Interactive evaluation has proven extremely useful in domains where finding a fitness-function is difficult or not possible at all. A human supervisor's "vague notion" of a concept has often led evolutionary search to astonishing results. This can be seen in the successful search for wasps' stigmergic construction behaviour (see Section 2.1) and the different flight formations of swarms (see Section 2.2). After Peter Bentley had produced shapes similar to butterflies and airplanes, the method of interactive evaluation has widely spread. Whenever aesthetics or solely creativity play a role, this approach comes in handy (further examples can be found in [5], [34], [39], [36], [22] and [18]).

The term "evaluation" refers to the part in an evolutionary search system that takes care of rating the individual's fitness. In Algorithm 2.1 evaluation is done by an external fitness function. Whenever the fitness rating of an evolutionary algorithm implements interactive evaluation, the whole search process may be called *interactive evolution*.

Considering objectives of interactive evolution, one can differentiate between those whose genome can directly be mapped onto the phenotype (e. g., Bentley's shapes [11] or Kwong's blob constructions [22]) and those whose genome encodes a procedure/behaviour that will result in an occurrence subject to time (e. g., Thomas' "developmental" art forms or the swarming behaviour studied by Jacob and Kwong).

In fact, it seems to be impossible to find a clear-cut definition of aesthetics. Though Lieckfeld postulates that useful and varied design, as a product of natural evolution, will be perceived as beautiful by most people [24], aesthetics - that's common sense - can not be formalised. Nevertheless, encountered once, there are a lot of attributes that

help to describe an object's aesthetics. In this context some keywords might help, such as *symmetry/asymmetry*, *pattern*, *proportion*, *harmony*, etc. [40].

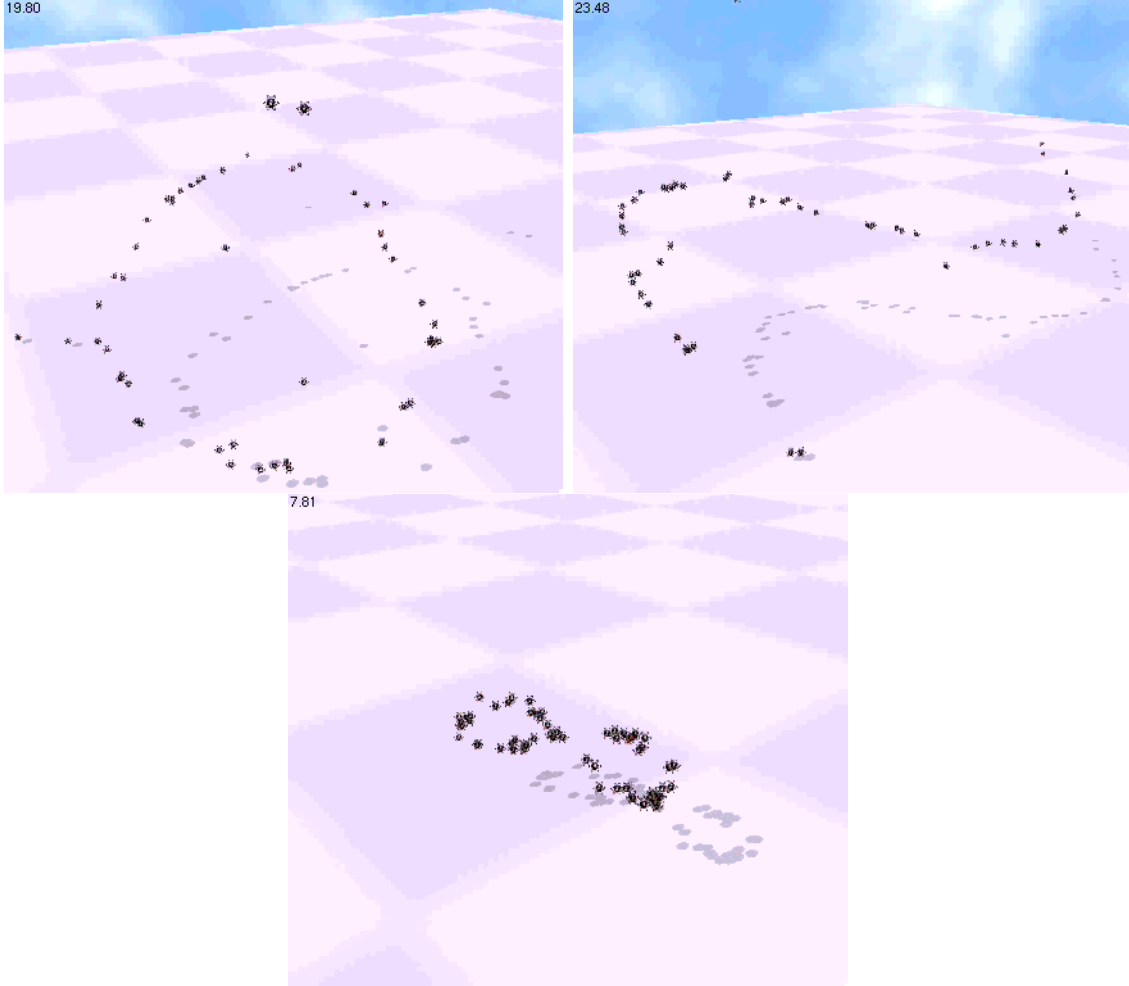


Figure 2.5: The first two images show the same swarm. Due to its tendency to switch from rings to sinuous lines, Jacob and Kwong named it "Big Ring Snake". On the right is a typical eight formation [23].

---

**Algorithm 2.1** Simple Genetic Algorithm, by Mitchell [27]

---

**Parameters:**

*Fitness*: A function that assigns an evaluation score, given a hypothesis

*FitnessThreshold*: A threshold specifying the termination criterion

$p$ : The number of hypotheses to be included in the population

$r$ : The fraction of the population to be replaced by Crossover at each step

$m$ : The mutation rate

**Returns:** The best hypothesis found

1: *Initialize population*:  $P \leftarrow$  Generate  $p$  hypotheses at random

2: *Evaluate*: For each  $h$  in  $P$ , compute  $Fitness(h)$

3: **while**  $[max_h Fitness(h)] < FitnessThreshold$  **do**

4:   Create a new generation,  $P_s$ :

5:   *Select*: Probabilistically select  $(1-r)p$  members of  $P$  to add to  $P_s$ . The probability  $Pr(h_i)$  of selecting hypothesis  $h_i$  from  $P$  is given by

$$Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^p Fitness(h_j)}$$

6:   *Crossover*: Probabilistically select  $\frac{r \cdot p}{2}$  pairs of hypotheses from  $P$ , according to  $Pr(h_i)$  given above. For each pair,  $\langle h_1, h_2 \rangle$ , produce two offspring by applying the Crossover operator. Add all offspring to  $P_s$ .

7:   *Mutate*: Choose  $m$  percent of the members of  $P_s$  with uniform probability. For each, invert one randomly selected bit in its representation.

8:   *Update*:  $P \leftarrow P_s$ .

9:   *Evaluate*: for each  $h$  in  $P$ , compute  $Fitness(h)$

10: **end while**

11: Return the hypothesis from  $P$  that has the highest fitness.

---

## 3 Approaching a constructive swarm

Several requirements have to be met to start artificial evolution. First of all, one has to make sure that the model of the evolution's objective can fulfil the expectations. In Section 3.1 general abilities of a constructive swarm are discussed. A model's details go hand in hand with its representation. As soon as the objective's representation stands, the genetical operators can be defined. Each of these nontrivial steps has been undertaken twice. Both approaches, together with remarks on the implementation and the simulation settings, are presented in the rest of this chapter.

### 3.1 Modeling a swarm that builds three-dimensional structures

Modeling the swarm in question comprises two aspects: To take something as an example for the swarm's actions and secondly, to decide how the swarm will look like, work, etc. The latter part might be better expressed with the term *designing*.

However, there are so many possibilities to choose from, that intuitive conclusions have to be made to come up with an overall model. A trivial example: It is not immediately clear, what consequences arose by using cubic blocks as construction elements instead of spherical ones. Spherical objects have a greater surface than cubic ones, while consuming the same volume in space. Such an implicit attribute might increase the frequency of collisions between agents and spherical particles. Therefore a decision for one or the other could lead to different structures or influence the agent's interactions. The dimensions of the simulated space, the amount of computed simulation seconds, the relation between an agent's and a particle's size and many other choices are subject to the programmer's intuitions.

### 3.1.1 Desirable results

In the course of this chapter, two concepts are presented that include a specific swarm model and an elaborate evolutionary system. Expectations may consequently be placed in three distinct areas:

1. The evolutionary system
2. The swarm model
3. A set of swarms with a specific constructional behaviour

The course of evolution is supposed to work effectively on the swarm's representation, in order to adjust the genepool into a certain direction. The swarm model's goal is to ensure that the swarm's functionality is complete in respect to the constructional task. To verify the overall concept's abilities, that is the framework comprising swarm model and evolutionary system, some construction swarms should be evolved and their output should be analyzed.

As the application for the developed "construction swarm" is not predefined, the properties of its output are not specified either. That is why it is attempted to satisfy the qualities of a structure (in Section 2.1) and to keep an eye on the aspects of creativity and aesthetics (see Section 2.3 on page 14) at the same time.

### 3.1.2 The relation between a swarm and its individuals

Natural swarms usually include individuals of different types. In the case that the individuals have the same abilities, they might automatically<sup>1</sup> specialise in an area to contribute to the swarm's needs in the best possible way. As far as this thesis is concerned, we only deal with uniform individuals (that is individuals of the same type) without any capabilities of specialization. This is considered as sufficient for a general approach, since it has already been shown that structures can emerge from swarms of such simple kinds (see Chapter 2.1).

---

<sup>1</sup>The more often a specific action is exercised by an individual - triggered by an external signal, the more it is reinforced to do it again. In this fashion experts are automatically cultivated according to the colony's/swarm's needs [12]



### 3.1.3 The individuals' basic abilities

From the given task we can derive some inherent abilities of the swarm agents: They should be able to construct and to wander in space. Basic decisions concerning the way of construction and movement are explained in the subsequent paragraphs.

By flying (instead of walking) the agents are able to reach a maximal number of points within a three-dimensional space. Thus flying is the least limiting traveling method (it offers the highest *degree of freedom*). One might argue that swarming implicitly means flying, but a bottom-up construction strategy, as in ant colonies, could have also been chosen - *swarm intelligence* as defined by Bonabeau et al. includes:

[...] any attempt to design algorithms or distributed problem-solving devices inspired by the collective behaviour of social insect colonies and other animal societies. [12]

The process of construction itself has undergone careful considerations, too. As framed in Section 2.1, there are a lot of inspiring examples for this process in nature. Ants and termites that carry and dispose objects (soil, wood-chips etc.) or wasps that produce carton as their very special building material. The latter one might be closest to our decision to let the agents create new construction elements appearing from nowhere. In this way, resources do not have to be organised<sup>2</sup> nor does their absence limit the creative abilities of the swarm.

If the swarmettes' flight<sup>3</sup> is neither predefined nor random, it demands permanent visual perception. Yet, triggering an agent's construction mechanisms only happens, if there are some stigmergical signals. Checking for the environment's structural configuration is guaranteed to make sense, as soon as the agent has collided with a particle already built. Of course, this is a simplification. It can be seen as the requirement  $distance\ d_{agentToParticle} = 0$  to get the construction process started. Yet, this is only a quantitative limitation of the general concept - if the swarmette is going to build a new particle at a specific location, it is supposed to be near that place anyways. So changing the environment is restrained to the events of collision.

---

<sup>2</sup>Organising resources is a large-scale task for swarm intelligences. It includes the search for resources, the construction of infrastructures and cooperative transport [12].

<sup>3</sup>Swarmette is another term addressing to the agents of a swarm.

### 3.1.4 Perception, data processing and action of the swarm individuals

The perception of a swarmette is limited to the senses of sight and touch. It might support the swarm's goals, if direct communication of the agents is possible (sense of hearing) and other senses are existent. But there are arguments for keeping the amount of sensory information on a minimal level. First of all it simplifies matters. Furthermore, in an artificial simulation environment, one can transfer certain stimuli onto signals that can be perceived by available senses. Originally *olfactory* (relating to the sense of smell, e.g. a pheromone) signals can be easily mapped onto visual marks. Even communication can happen indirectly and by means of qualitative or quantitative stigmergy (as seen in Section 2.1).

The behaviour of an individual can be determined by a set of rules, a neural network or any other inference system considering the input information and implying an output action. In addition, memory could influence the inferred action. In the individual's memory things could be stored such as: recent construction locations, attributes of the last particle built, the number of built particles or whole lists of the last events etc. This might lead to an agent's specialization, for example it might become responsible for the construction of whole modules of an architecture. Still, for a start it is already a challenging task to come up with a whole swarm that is specialised in building only a part of an architecture. Finally, the individual's own physical situation can play a role in its building behaviour. Just consider an agent that is supposed to start building a roof, after it has reached a certain height. In this case, allowing the agent's own situation to influence its actions seems reasonable.

The possible actions of a swarm agent are very few. It can adjust its flight to its swarm-mates in the neighbourhood and build or destroy construction elements. Once the decision has been made to create a swarm of flying agents, their movements are coordinated in accordance with the model of swarming, introduced in Section 2.2.

Of course, perception, input processing and action of a swarm individual must be connected: Input information has to be processed by an inference system which yields an action. The following sections offer two approaches which differ from each other in most of these respects (Table 3.1 contrasts both approaches by keywords).

The succession of discussed topics may be explained in a few words: The swarm's task to create 3D structures claims for a set of basic abilities such as moving in space and building particles. Each swarm agent has to be equipped with the necessary fundamental operators which has to be done first and serves as basis for further discussions. An agent usually consists of three main parts: Its actuators, its perception and an internal system that connects these modules in a particular way. After predefinition of the actuators, the sensory features have to be specified. A simulation is set up only for the purpose to develop a creative swarm. The designed world offers several sources from which the agent could gain its knowledge. An agent's perception will determine its area of influence and has to be done with careful consideration. Development of a proper inference system has been chosen as third and last step of designing the agent. The type of information that is processed and the kind of information that represents the agents' actions influence the decision for a certain system that connects a swarmette's input and output. Of course, the manner of knowledge processing is constitutive for the agent's behaviour as well. After these steps have been carried out, the swarm agent's model is poised to be evolved to build 3D structures. Therefore the final steps of each approach's presentation outline the evolutionary process which will lead to a proper creative swarm.

<b>Connectionist Approach</b>	<b>Rule-based Approach</b>
<i>Quantitative stigmergy</i> Usage of thresholds, processing of continuous signals	<i>Qualitative stigmergy</i> Clear cut input data about the agent's surrounding structure
<i>Smoothness</i> Particles are overlapping, colourful and vary in size	<i>Plainness</i> Non-overlapping, uniform particles
<i>Interactive evolution</i>	<i>Automatical orientation towards a given 3D object</i>

Table 3.1: Contrasting the general differences of the two approaches

## 3.2 Connectionist approach

The first approach is based on an implementation by Jacob called “Towers” in which the agents start piling up particles as soon as they have collided with another particle or the ground. Of course, this quickly leads to immense computational costs. With no limitation to this behaviour, shortly many thousands of particles flood the simulation and therefore the computer’s main memory. Nevertheless, the program influenced this connectionist approach’s general character and implicitly answered some questions about the details of the construction process. Therefore, the following paragraph draws a more detailed picture of the “Towers” simulation (Figure 3.1 shows two sample images of the “Towers” simulation).

Think of a three-dimensional environment, (graphically) consisting of the ground and the sky. In this environment a number of spheres is swarming around. Indeed, these spheres represent the swarm individuals. They are remnants of the earlier works by Klein and Spector [35]. Jacob has changed the simulation in such a way that each time an individual collides with an object, a new particle is added on top. The created particles are uniform, white spheres. An agent collides with the ground and the particles already built. In “Towers” collision of particles themselves is not tested at all. Consequently, the newly created shapes overlap with older ones in most cases. This continuity of the construction looks natural and smooth and it is therefore a good leading idea for the (first) overall approach.

In the upcoming subsections an elaborate model is presented that extends the “Towers” agents’ sensory capabilities and connects their constructional behaviour to the incoming information. The further developed “connectionist” swarm agents are equipped with a much more complex inference system that determines their doings. In order to train the agents’ intelligence, the means of interactive evolution are applied which is also part of the overall approach.

### 3.2.1 The individuals’ construction operators

An agent collides with a particle or the ground. Instantly, the agent’s internal system decides how to react. The possible reactions range from “do nothing” to “do all the things you can”. The latter conglomerate of actions consists of *destruction* of the particle the

agent has collided with and of *creation* of a new one. Whereas destroying an existing particle is either done or not, the creation of a new particle needs some more deliberation. For once, there are the attributes of the new particle that have to be determined: Size and colour (a construction element is a sphere that can vary in size and colour). Further on, the agent can place it anywhere around itself.

The whole simulation setting's granularity is only limited by the internal precision of floating point numbers. For that reason, the parameters of the agents' actions are continuous, too. Colour is defined after the RGB model (which means that each objects' colour is a combination of the three colour-channels red, green and blue) and consists therefore of three values from  $[0, 1]$ . A particle's size ranges from  $[0, 3]$ . The new particle is placed right next to the agent. As the agent itself is represented as a sphere, any direction can be considered.

In total nine numbers (two boolean, three integer and four floating point values) are needed to describe the agent's construction behaviour. This happens, as mentioned before, as soon as a collision with a particle or the ground occurs.

Statics do not play a role, that is why an agent may build right into the air. The original "smoothness" of the Towers simulation is kept: Particles may still be built into each other, so that they overlap. This aspect is even intensified by the construction apparatus that attaches continuous attributes to the newly built particles.

Action	Possible Values	Parameters
Destroy old Particle	Yes/No	None
Create new Particle	Yes/No	Colour, Size, Location

Table 3.2: Set of actions of a connectionist agent

### 3.2.2 Behaviour determined by weighted input

Except for the models that we have seen in Chapter 2, nothing indicates what information is useful for a swarm individual to build structures. For that reason every directly available information is passed to the swarmettes. Moreover, calculations are made to produce some additional data. Table 3.3 gives a complete overview of the swarm indi-

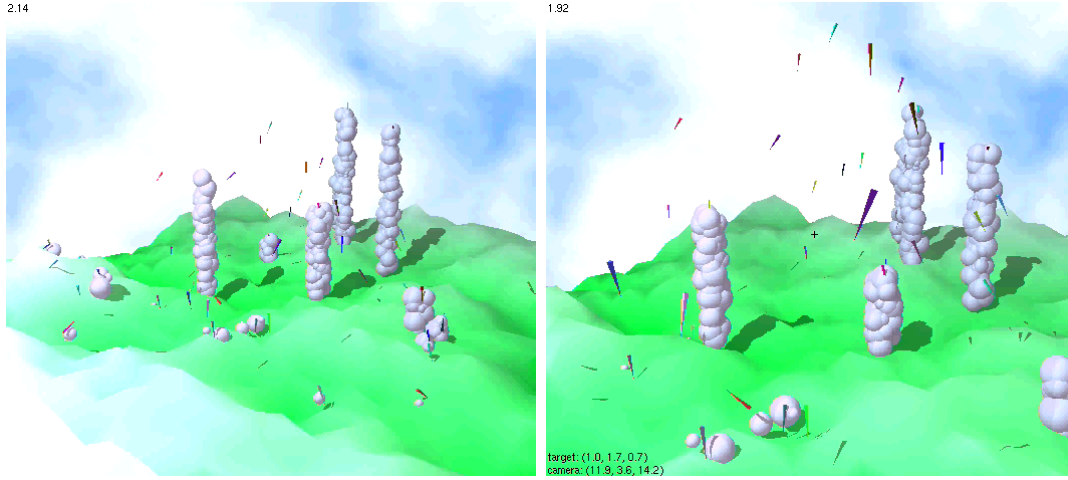


Figure 3.1: These are sample images of Jacob’s “Towers” simulation. The colourful arrows are the swarm individuals. As soon as they reach the ground they start piling up spheres.

vidual’s accessible data. The necessary objectives to ascertain the input data are the agent itself, the particle it has collided with and its surrounding particles.

The agent is supposed to derive its actions (Table 3.2) from these input values. Processing this huge amount of data is done by a *one-layer neural net*. The 29 numbers (sum of all numbers in the Representation column of Table 3.3) are connected to nine *computational units*. According to the kind of output needed, *sigmoidal* and *perceptron* units are used (a detailed introduction to neural networks can be found in [32]). Both types consider the weighted sum of all input values  $x_i$ ,  $i \in \{0, \dots, n\}$ :

$$net_i = \sum_i w_i x_i , \quad (3.1)$$

whereas  $w_i$  is the weight of the edge that connects input  $x_i$  with the unit.

If the equation  $net_i \geq 0$  holds, a perceptron unit yields 1, and 0 in the other case. Such a perceptron with an *extended input vector* assumes that the first input value equals minus one,  $x_0 = -1$ . The first incoming edge’s weight,  $w_0$ , can then be seen as a *threshold* value,  $\Theta$ . Due to its binary output, it suits perfectly for boolean decisions, like those about creation and destruction in Table 3.2.

Description of the information	Representation
<i>About the agent's situation</i>	
The agent's velocity at the time of the impact	3D Vector
The agent's location	3D Vector
<i>About the particle the agent has collided with</i>	
The old particle's location	3D Vector
The old particle's color	RGB-triple
The old particle's size	Scalar
<i>About the neighbouring particles</i>	
The number of neighbours	Scalar
The neighbours' minimal/maximal and average colour	One RGB-triple each
The neighbours' minimal/maximal and average size	One scalar each
The neighbours' minimal/maximal and average height	One scalar each

Table 3.3: The connectionist agent's available information (16 floating point and 13 integer values)

A sigmoidal unit (also called *sigmoid*) yields an output value in the interval  $[0, 1]$  by computing the function:

$$s(net_i) = \frac{1}{1 + e^{-net_i}} \quad (3.2)$$

A sigmoid delivers the proper value for each of the swarmette's construction parameters (location, colour and size of a new particle). Figure 3.2 shows the connections of the "behavioural net" of the swarm's agents. The output depends on the input values and the weights along the edges. Put in other words: The weights of the net are responsible for what an agent does in each situation.

The swarm's flocking behaviour results from the interplay of another class of actions: E.g. the alignment of an agent's direction relative to its neighbours as presented in Section 2.2. Depending on the weight that is assigned to each of these actions, a distinct flight formation of the swarm results. Table 3.4 shows a parameter set that causes a swarm to "flock normally", which means that the agents fly in small groups and it comes often about, that single members of the swarm wander off.

Swarming Parameter	Value
SPACING_CONSTANT ( $c_1$ )	3.0
WORLD_CENTER_CONSTANT ( $c_2$ )	2.0
VELOCITY_CONSTANT ( $c_3$ )	10.0
CENTER_CONSTANT ( $c_4$ )	2.0
WANDER_CONSTANT ( $c_5$ )	4.0
MAX_VELOCITY ( $V_{max}$ )	15
MAX_ACCELERATION ( $A_{max}$ )	15
CRUISE_DISTANCE ( $d$ )	0.4

Table 3.4: These values determine the connectionist agent’s swarming behaviour (after [35]).

The connectionist swarm agent has now the capabilities to swarm in space and react on its environment to build structures. The determination of the individual’s behaviour lies completely in the network’s weights. Each of the nine computational units is connected with all 35 input values. Consequently, the whole network can be seen as a *matrix*, comprising nine *vectors*, each consisting of 35 weights (in total that is  $9 \times 35 = 315$  single values).



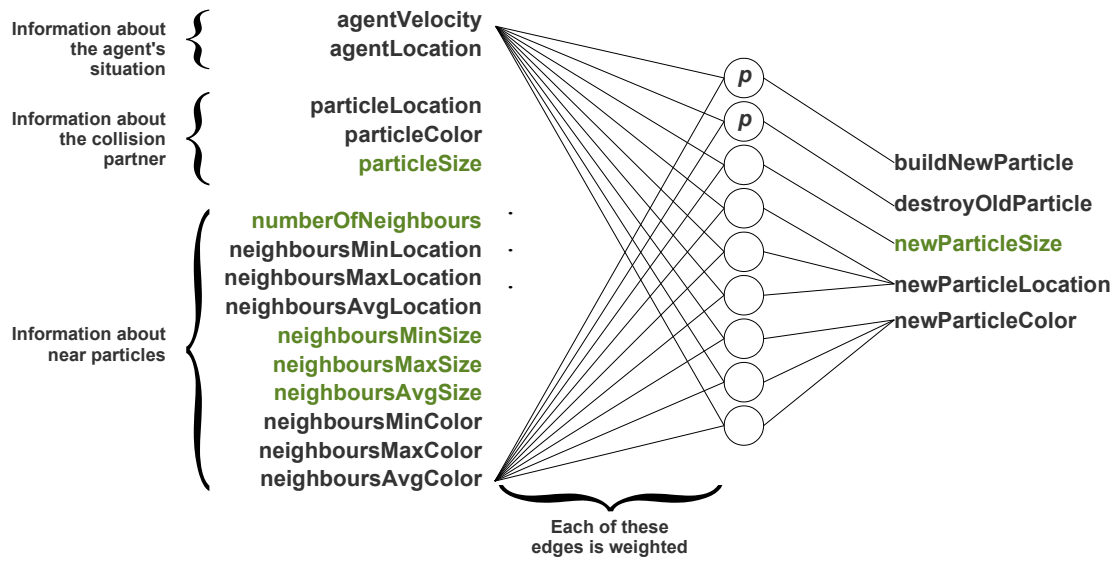


Figure 3.2: The “behavioural net” of a swarmette: The left side shows the available input data. In the middle part there are the computational units - perceptrons are denoted with a “p”. The output on the right, which depends only on the weights along the edges, is interpreted as the individual’s set of actions. Most terms represent vectors, except for the green ones. Each computational unit “fires” one scalar.

### 3.2.3 Interactive evolution

After the layout of the agent's abilities is made, the final step is to find configurations of the agents' behaviour that induce the creation of interesting structures. Section 2.3 has introduced the means of interactive evolution, which apparently answer the needs for the faced task.

A swarm's genotype is represented by the described  $9 \times 35$  matrix of the behaviour network. The according phenotype is the construction that is built by the corresponding swarm. Interactive evolution is realised by repetition of the following steps (this process is illustrated by Figure 3.3):

1. Generation of a population of genotypes (at first randomly, in the next iterations by applying the genetic operators which are described in Subsection 3.2.4)
2. Non-graphical computation of the phenotypes
3. Simultaneous visualization of the population of constructions
4. Comparison and rating of the constructions by a supervisor (with values from 0 to 9, where 0 indicates the worst and 9 the most satisfying result).
5. Returning the genotypes and the attached ratings to the genetic operators that are used in step 1

As in the simple genetic algorithm (Algorithm 2.1), the first population of swarms is created randomly. The weight matrices that determine each swarm's behaviour are initialized with random numbers from  $-5$  to  $+5$ . This initialization guarantees that processing a single input value can cover the whole range of achievable outcomes: 1 is an *operational* value of all given input data. Equation 3.1 with  $x_i = 1$  and  $w_i \in [-5, +5]$  leads to an outcome in the interval  $[0, 1]$  for the sigmoidal unit (Equation 3.2) and enables the perceptrons to fire 0 or 1.

### 3.2.4 Genetic operators

The schemed interactive evaluation furnishes selection with adequate values to choose individuals from the current population and transfer them to the next generation. The

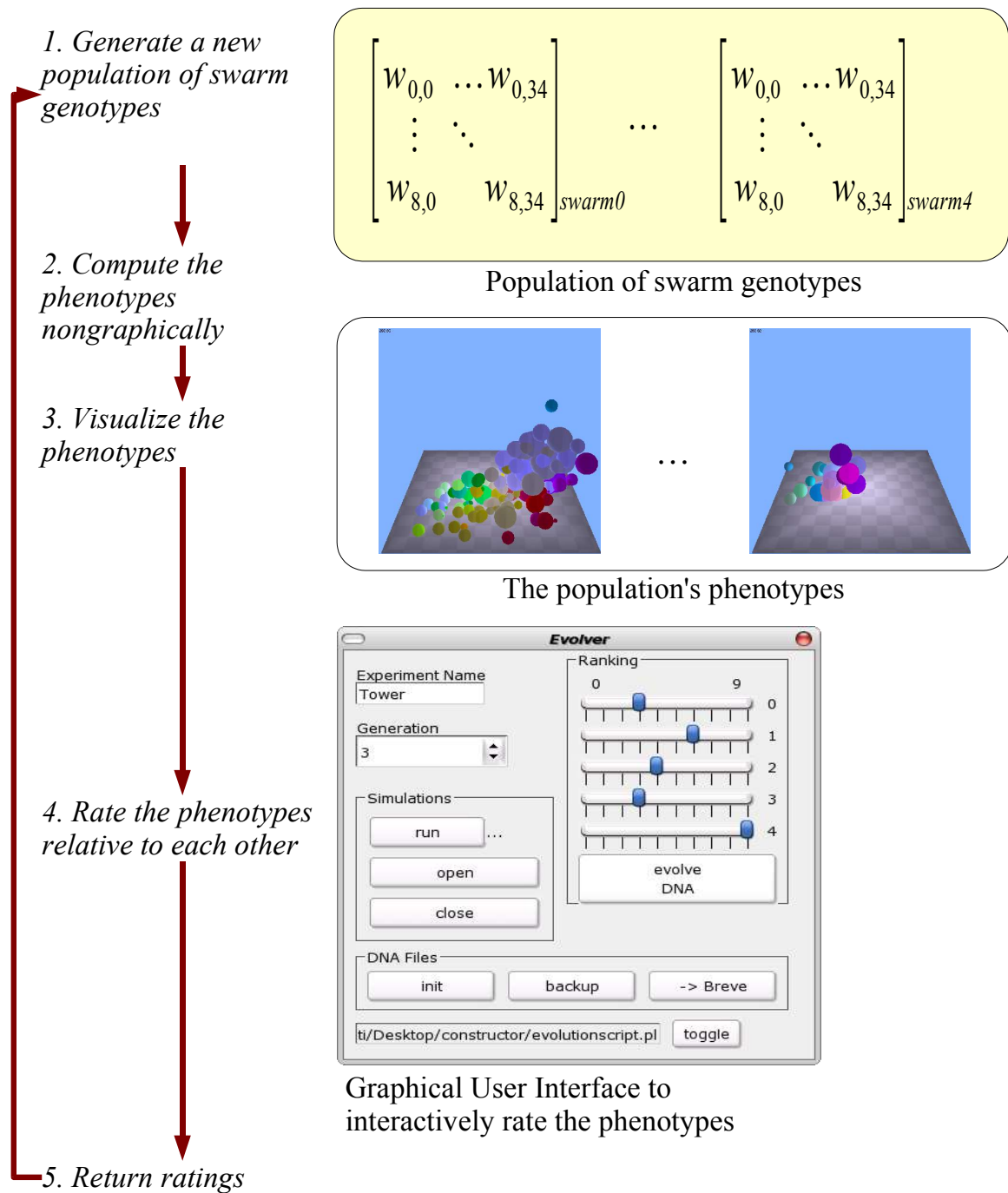


Figure 3.3: The course of interactive evolution implemented to evolve a “connectionist” swarm.

rest of the genetic operators work on the genotypes themselves, represented as weight matrices. This subsection describes the implemented genetic operators and supplementary functions.

**Rank-based selection:** The individuals are ranked in accordance with their ratings. To each rank a (fix) probability of selection is attached. Rank-based selection avoids *crowding* [27], which would mean that one individual is overly predominant in the upcoming generation. Fitness proportionate selection, for example, has a great tendency to choose only the best individual, if its fitness is extremely high compared to the others' ratings. For a small population size crowding can happen within a few generations. Due to that reason, rank-based selection is used.

**Crossover mask:** This mask tells the crossover-operator (see Section 2.3) which part of the genome should be taken from which ancestor. It is a binary string: 0 indicates that an *allele* (a part of the genome of a defined length) is inherited from ancestor *A*, a 1 signals that the allele originates from parent *B*. Here, the crossover mask is a random concatenation of zeros and ones.

**Crossover on matrices:** Merges parts, in particular vectors, of the behaviour network matrices from two ancestors according to a given crossover mask. It produces two new individuals.

**Crossover on vectors:** Mixes the values of two (weight-)vectors according to a given crossover mask and thereby generates two new vectors.

**Mutation:** Changes a value with a given probability. A threshold is declared that has to be surpassed, in order to execute mutation at all. If a random variable  $X$  is greater or equal than a mutation threshold  $\Theta_m$ ,

$$X \geq \Theta_m \tag{3.3}$$

the mutation operator updates the original value  $v_0$  to the new value  $v_1$  by adding or subtracting the term:

$$\Delta v = 1 - e^{-x^2} \quad \text{with a random } x \in [0, 1] \tag{3.4}$$

$$v_1 = v_0 \pm \Delta v \quad \text{with } v_0 \in [-5, +5] \tag{3.5}$$

(increase and decrease of the original value are equally probable)

The idea to mutate all values of every individual is taken from [16]. However, the mutation threshold  $\Theta_m$  (in Equation 3.3) reduces the chances of an actual value change.  $\Theta_m$  defines the percentage of mutation on the next generation's total genetic information (also *gene pool*). The update function (Equations 3.4 and 3.5) ensures a smooth alteration because minor changes are more common (see Figure 3.4).

The set of presented genetic operators can be understood as a special implementation of the standard operators, introduced in Section 2.3. If not mentioned differently, it is fully complied with the simple genetic algorithm (Algorithm 2.1). Details of the system's settings and the interplay of the evolutionary algorithm, simulation runs and the realization of interactive evaluation are presented in the next subsection.

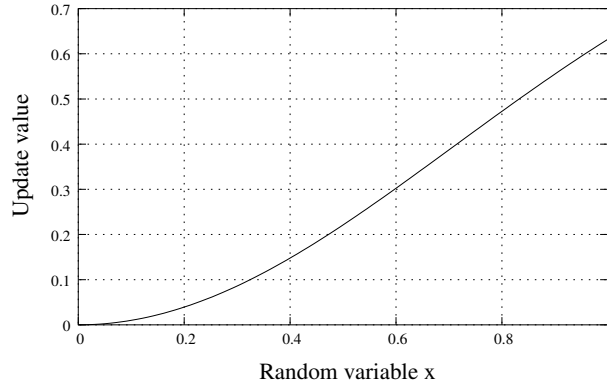


Figure 3.4: The update  $\Delta v$  that is added or subtracted from the original weight value depends on a random value  $x \in [0, 1]$ . It can be seen that minor changes have higher probability.

### 3.2.5 Implementation details

There are only a few parameters that determine the evolutionary process and the swarm simulations to compute the phenotype. They are listed in Table 3.5.

The phenotype is computed with a simulation environment called **Breve** which is developed by Jon Klein [21]. Breve interpretes an agent-based implementation of the

<i>Simulation parameters</i>	
Number of individuals per swarm	20
Spatial dimensions	$100^3$
Size of an agent	1
The agent's neighbourhood radius	3
Maximal size of a particle	3
A particle's neighbourhood radius	3 times its own size
Simulated seconds	300
<i>Evolution parameters</i>	
Population size	5 swarms
Crossover rate	0.5
Mutation threshold $\Theta_m$	0.9

Table 3.5: Default parameters of the connectionist approach's implementation

swarm (in the language **Steve**). To keep the swarms' outcomes comparable, each process is run until 300 simulated seconds have passed. During this time interval visualization is switched off. For an additional computational speed-up processing the population is distributed onto five computers. After the simulations have run and their output has been stored, the results are transferred to the most powerful machine. There the visualization takes place and a supervisor can rate the swarms' constructions. Interactive evaluation is preferably done with an easy-to-use graphical interface. The Qt-library by Trolltech [37] has been used to create a simple GUI panel that accepts the following tasks and calls the proper functions of a PERL-script (the panel can be seen at the bottom of Figure 3.3):

1. Initialization of a population
2. Import of an existing population
3. Creation of a new generation based on input ratings
4. Start and stop of the simulation process

5. Visualize the current result (the successful completion of a simulation run is indicated)

### 3.3 Rule-based approach

The second approach adheres to the means of qualitative stigmergic nest-building presented in Chapter 2. It can be seen as the absolute opposite to the connectionist approach. Although, the simulated world is still “continuous”, in the sense that it is not divided by a three-dimensional grid, this attempt ignores the connectionist approach’s postulated “smoothness” completely. In particular it deviates in the following points:

- Particles may not overlap anymore
- Construction elements are uniform: They are all of cubic shape and share the same size and colour
- The agents’ construction behaviour is determined by a set of rules instead of a neural net
- The input information is limited to:
  - The necessary data for the swarming behaviour
  - The local structural configuration on which the building behaviour depends
- Interactive evaluation is replaced by measuring the fitness through the approximation of a given 3D structure
- The genotype includes the set of parameter values that determine the swarm’s flight formation

Conceding the construction elements to overlap finds its legitimation as soon as one thinks of glueing objects together. If objects are supposed to be piled up or consist of solid material, overlapping is out of the question. The swarm’s implementation implicitly asks for a decision in favour of the one way or the other. There are good arguments for both methods. Overlapping is realised by the connectionist approach. For studying

as many aspects as possible, the rule-based approach implements the construction with separated particles<sup>4</sup>.

A cube is the simplest shape of an object, which is to be piled up. Since an agent does not consider anything for the construction process, except the local structural configuration, a uniform cube size makes sense, too. The agent's environment is formed by a conglomerate of cubes. Since many small cubes can form a bigger one, the distinction between their size in the individual case would not make a difference. Figure 3.5 illustrates the interplay between a spherical agent and cubic construction elements.

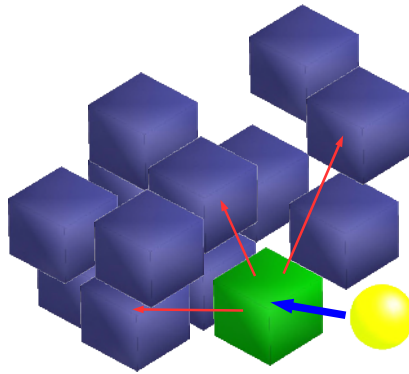


Figure 3.5: An agent, represented by a yellow sphere, is going to collide with the green cubic construction particle (the agent's velocity is indicated by the blue arrow). The agent's behaviour depends on the structural configuration that surrounds the collision partner. In the illustrated case red vectors point to the locations that will be checked for particles by the agent.

To keep the construction process simple, marks, such as attaching different colours to the single particles, are not used either.

It turns out that each run of a simulation, that is guided by interactive evaluation builds upon a certain idea. At first one might, for example, try to achieve a construction that reaches very high. During the course of evolution the supervisor might run into an unforeseen though interesting structure that inspires his/her objective's notion. However, when starting the simulation, the supervisor must have in mind a reasonably

---

<sup>4</sup>To avoid overlapping bricks, a test for collision must be performed before a new particle is built. One can easily revert to overlapping objects by deactivation of this collision detection.



explicit conception. If one is able to map some attributes of the imagined structure onto a three dimensional construction (such attributes can be height or a general shape, etc), the provision of orientation towards an object comes in handy.

The agents' way of swarming might also influence the coordination of their building behaviour. To concede the facility of adjusting its flocking behaviour to a given task, the destining set of swarming parameters is included into the genotype. Thus, the swarm's flight undergoes the same evolutionary process as its building behaviour.

In total this approach cherishes the idea that complex structures are realised by regarding qualitative stigmergy in tandem with adoption of the swarm's flocking behaviour. Instead of interactive evaluation, the approximation of a given three dimensional structure takes over the task to guide the evolution of a suitable swarm.

### 3.3.1 The individuals' operators

The swarm's flocking behaviour results from a set of parameters. Pursuant to Reynold's swarm model and its extensions (Section 2.2) the agents are equipped with visual senses to see their neighbours. Assume  $\vec{d}_{si}$  as the vector between a swarmette  $s$  and another individual  $i$  of the swarm. Every individual  $i$  is in the neighbourhood  $N$  of a swarmette  $s$ , if the absolute value of  $\vec{d}_{si}$  is within the swarmette's radius of perception  $r$  and the angle  $\alpha_{si}$  between the direction of  $s$  and the location of  $i$  is within some range  $[0, 2]$ .

$$\forall i \in N_s | (\|\vec{d}_{si}\| \leq r) \wedge (\alpha_{si} < 2) \quad (3.6)$$

In each time step of the simulation, the swarmette's velocity  $\vec{V}_{velocity}$  is updated with an acceleration vector  $\vec{V}_{acceleration}$ :

$$\vec{V}_{acceleration} = \sum_{j=0}^5 w_j \vec{V}_j \quad (3.7)$$

$w_j$  ,  $j \in [0..5]$  are the weights of the distinct tendencies during the flight.

The six urges are:

$\vec{V}_0$ , **Center**: Yields the vector towards a fixed location within the simulation environment or to an agent's location. If no center is defined,  $V_0$  is zero.

$\vec{V}_1$ , **Separation:** Implemented as explained in Section 2.2:

$$\vec{V}_1 = - \sum_{i \in N_s} \frac{\vec{d}_{si}}{\|\vec{d}_{si}\|^2} \quad (3.8)$$

$\vec{V}_2$ , **Alignment:** Implemented as explained in Section 2.2, referring to the neighbour's velocity  $\vec{v}_i$ :

$$\vec{V}_2 = \sum_{i \in N_s} \frac{\vec{v}_{si}}{\|\vec{d}_{si}\|^2} \quad (3.9)$$

$\vec{V}_3$ , **Cohesion:** Implemented as explained in Section 2.2, referring to the neighbour's location  $\vec{l}_i$ :

$$\vec{V}_3 = \sum_{i \in N_s} \frac{\vec{l}_{si}}{\|\vec{d}_{si}\|^2} \quad (3.10)$$

$\vec{V}_4$ , **Ground:** Is responsible for the general tendency towards the ground.  $height_s$  entitles the swarmette's height and  $height_{max}$  is the maximum height of the simulation environment.

$$\vec{V}_4 = \begin{pmatrix} 0 \\ \frac{1}{2}height_s/height_{max} \\ 0 \end{pmatrix} \quad (3.11)$$

$\vec{V}_5$ , **Random:** A normalized random vector.

$\vec{V}_0$  offers a new potentiality.  $\vec{V}_0$  refers to a center of the swarm which is not implicitly given but has to be explicitly announced. An individual has to declare a specific location or itself as center of the swarm. If  $w_0$  is sufficiently high (compared to the other weights), a tendency towards the selected goal will appear. In this way one can say that all the agents have a common leader (if an agent has declared itself as center of the swarm) or, in the other case, that the swarm focuses onto a certain region. Another action comes along with setting or changing the swarm center: An agent may recant the declaration of a center, so that former announcements have no influence on the single agent's acceleration anymore.

As in the connectionist approach, the agents have an infinite number of construction elements at their disposal. Once again the attention is turned to the resulting structure

and explicitly not to meet the demands of applied sciences. Hence the agent's basic abilities remain creating and destroying a new particle. The individual's functionality is elaborated as follows.

A brick is built at a location relative to the spatial coordinates of the particle the agent has collided with. This relation is stated by a *direction vector*  $d$ . There is a set  $D$  of given direction vectors according to the basic points of the compass (in detail these are  $\vec{d}_{North}$ ,  $\vec{d}_{South}$ ,  $\vec{d}_{East}$ ,  $\vec{d}_{West}$ ,  $\vec{d}_{above}$ ,  $\vec{d}_{below}$  and  $\vec{d}_{here}$ ). The absolute values of these standard direction vectors are normalised to the bricks' size, so that direction and distance of  $\vec{d}_{above}$  point from one particle exactly to its upper neighbour's center.

There are two types of destruction methods. The agent may destroy the particle it has collided with. Another method is to destroy a particle at a relative distance. The location of the particle in question is computed by addition of a vector  $\vec{d}_{destroy}$  to the collision particle's coordinates  $\vec{p}_c$ . Of course, if there is no particle at  $\vec{p}_c + \vec{d}_{destroy}$ , destruction does not take place.

The rule-based agent's actions are summarized in Table 3.6.

<i>Class</i>	<i>Action</i>	<i>Parameter</i>
1	Create new particle	Vector (relative to the the collision particle's location)
2	Destroy the collision particle Destroy a remote particle	None Vector (relative to the the collision particle's location)
3	Set center to itself Set center to a specific location Cancel swarm center	None Vector (relative to the the collision particle's location) None

Table 3.6: The rule-based agents' actuators

### 3.3.2 Behaviour determined by “if-then-rules”

The agents’ behaviour is determined by a set of Rules  $R$ . Each rule  $r \in R$  has the form:

$$c_0 \wedge c_1 \wedge \dots \wedge c_n \rightarrow a \quad (3.12)$$

Whereas each condition  $c_i$  is internally represented as a vector  $\vec{p}_i$ . A condition is fulfilled, if a brick is found at  $\vec{p}_i + \vec{p}_c$  ( $\vec{p}_c$  denotes the location of a particle the swarmette has collided with). The rule’s consequence  $a$  consists of an action (one of Table 3.6) and a vector which is sometimes used as a parameter.

Since there is no lattice that divides the three dimensional space into discrete unit cubes, the way to ascertain a brick’s existence at a given location is not straightforward. One method to realize this test is checking whether a collision will occur, if a cube of minimal size is built at the given coordinates. Figure 3.6 shows this procedure.

The actuator on the rule’s right is only triggered if all the conditions on the left side are satisfied. Once a collision occurs, each rule  $r \in R$  is tested and if applicable its consequent action is executed.

The number of rules  $|R|$  and the length of a rule (which is equivalent to the number of conditions and therefore denoted as  $n_{conditions}$ ) are given. The higher the number of conditions the lower is the chance that a rule is complied with. In order to leave the rules easily applicable, the number of conditions is limited to five. The number of rules is set to 20.

When initialized, to each condition a vector  $\vec{v}_i$  is (randomly) attached that is element of the set of basic directions  $D$  introduced in Subsection 3.3.1. The same holds for the vector that is part of the rule’s consequence. The action itself is found in two steps:

1. Choose the general class of action (there are three classes of actions: those related to creation, to destruction and to the swarm center, see Table 3.6).
2. Take one of the actions available in the specific class.

This section has clarified, how processing the perceived data is linked to distinct constructional actions presented in Subsection 3.3.1.

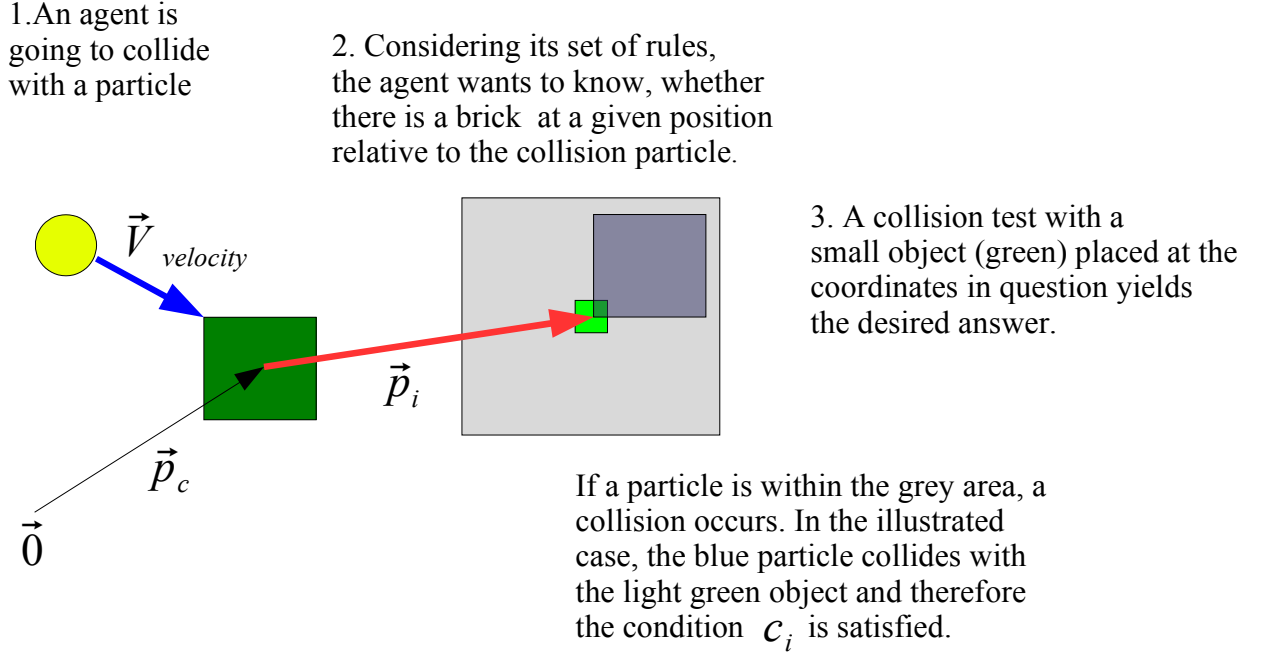


Figure 3.6: To analyze the surrounding structural configuration, the agent tests, whether there are bricks at certain positions or not. The figure illustrates the following case: An agent  $s$  collides with a particle  $c$ . One of the rules of  $s$  checks for a brick at  $\vec{p}_i$  relative to the position  $\vec{p}_c$  of the collision particle.

### 3.3.3 Approximation of a given 3D structure

Perception, data processing and a set of actions of the rule-based swarm agent are outlined in the previous sections. The next important task is to find a set of rules and flocking parameters that lead to structured constructions. Automation of the evolutionary search for an appropriate genotype requires a mathematical way to measure a swarm's fitness. In order to guide the search, the difference between a swarm's construction ( $construction_{swarm}$ ) and a given three dimensional structure ( $construction_{predefined}$ ) is used as the swarm's fitness rating.

The fitness function's prototype looks like this:

$$fitness(construction_{swarm}) = construction_{swarm} \cap construction_{predefined} \quad (3.13)$$

The given construction is composed of several cubes, specified by position and range. Since the difference of two 3D objects is equivalent to their intersecting volume, the set of built bricks is tested against the set of given cubes for intersection. Based on Algorithm 3.1 it is simple to compute some values that help to implement a proper fitness function.

$C$ , “**Covering Volume**” represents the summed intersections of built and predefined particles.

$\neg C$ , “**Non Covering Volume**” is the volume of the swarm’s construction that does not intersect with the predefined structure.

$F$ , “**Fitness Object Volume**” , the total volume of the given structure.

Equation 3.13 can now be implemented as follows:

$$fitness(construction_{swarm}) = \frac{C}{F} - \frac{\neg C}{F} \quad (3.14)$$

The term  $\frac{C}{F}$  is added to the fitness. It reaches its maximum when  $C = F$ , which is the case if the whole given structure is rebuilt by the swarm. But this case does not exclude the eventuality that the swarm might have constructed something totally different and that the predefined structure is only covered incidently. To ensure that the swarm’s construction is as near to the given 3D objects as possible, any outgrowth is rated negatively by the second term of Equation 3.14,  $\frac{\neg C}{F}$ . The more particles are built that do not contribute to the given structure’s approximation, the lower is the fitness of the swarm. Instead of direct subtraction of the volume of the misplaced particles, a softer penalty is imposed. The volume of the unwanted bricks is set into relation to the fitness structure’s volume. In this fashion the punishment is small as long as the swarm creates fewer particles than necessary to fill up the given structure. This gives the swarm an impetus to construction in general. However, if the built structure grows rampant, the penalty will ruin the swarm’s fitness.

### 3.3.4 Genetic operators

This section describes the functions used to create new generations in the evolutionary process. In general, all steps are implemented according to the simple genetic algorithm (Algorithm 2.1). However, the genotype’s representation, that is the set of construction

---

**Algorithm 3.1** Volume of Two Cubes' Intersection

---

**Parameters:**

$\vec{a}, \vec{b}$ : The cubes' positions

$range_a, range_b$ : The range is half the size of a cube

**Returns:** Intersecting volume of cubes  $a$  and  $b$

$$\vec{d}_{min} = \vec{d}_{max} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \text{ \{these vectors describe the intersecting box\}}$$

**for all**  $i \in [0..2]$  **do**

**if**  $(a_i - range_a) < (b_i - range_b)$  **then**

$$d_{min,i} = b_i - range_b;$$

**else**

$$d_{min,i} = a_i - range_a;$$

**end if**

**if**  $(a_i + range_a) > (b_i + range_b)$  **then**

$$d_{max,i} = b_i + range_b;$$

**else**

$$d_{max,i} = a_i + range_a;$$

**end if**

**end for**

$$volume = \prod_{i=0}^2 d_{max,i} - d_{min,i};$$

Return  $|volume|$ ;

---

rules and flocking parameters, needs some adjustment of the genetic operators. Details that are kept open by Algorithm 2.1 are stated, too.

**Crossover** Although a whole set of crossover modes is implemented (*random*, *equal*, *one point* and *multi-point crossover masks*, as described by Mitchell [27]), it is sufficient to confine to one of them. For the simulation a *two point* crossover mask is chosen, so that each of the two offspring owns one part of one and two parts of the other ancestor. If there exist any dependencies within the agent's genotype (e. g., a constructional rule that makes only sense with another one within the same set), its partitioning into three parts may very likely conserve them.

There are several levels in the genotype's hierarchy. The genotype is split in construction rules and flocking parameters. Again, the set of construction rules consists of a number of rules that own lists of conditions. On each of these levels the crossover operator could come into action: On rules, sets of rules, sets of parameters or on the whole genotype. Of course, the latter kind is taken to generate the offspring for a new generation. In this fashion, the offspring inherit flocking parameters and single constructional rules according to the crossover mask. The offspring's number of rules is limited to the smallest order of the ancestors' rule sets. By this means it may happen that the average size of the swarms' rule sets decreases in the course of evolution. A small set of rules and still good performance corresponds with *Occam's razor* which states that [27]: "the simplest hypothesis is the best".

**Mutation** At first the mutation operator checks whether it should alter the given object or not - according to a given mutation rate. If the decision is made in favour of alteration, a number or vector, smaller than a given mutation distance, is chosen and added to or subtracted from the original value. Whenever the resulting values leave an interval (defined by a lower and an upper bound), they are trimmed to the next boundary. This procedure is applied on every genotype of the new generation. There is no special motivation for this mutation procedure. The mentioned boundaries ensure that the evolved parameters make sense and the mutation distance defines the procedure's maximum effect.

#### 3.3.5 Implementation details

Table 3.7 lists the default parameters of the evolutionary system and of the simulations which ascertain the phenotypes' fitnesses. In opposition to the first approach (Section 3.2) a supervisor only has to adjust the parameters of Table 3.7, provide a 3D structure that allows to compute a swarm's fitness and start the evolutionary process.

The evolution is executed by iteration of two steps:

1. Iterative computation of all swarm phenotypes along with their fitness
2. Application of selection, crossover, mutation as described in 3.3.4



The structure which guides the evolutionary process is read from a file. The genotype (also *DNA*) of each swarm along with its achieved fitness are stored in a protocol file. The evolutionary process can be resumed, if it had to be stopped. The  $k$  fittest genotypes are saved automatically in a separate file. A multi-start hillclimber (as described in Chapter 2 on page 12) can be launched by supplying the program with a file consisting of an appropriate set of genotypes. The file which contains the  $k$  best swarm genotypes can be directly used for this purpose.

Triggering the swarm's creative behaviour depends on at least one brick (otherwise the set of constructional rules will not even be considered, see 3.3.2). Due to this fact, a "seed"-file is read, which tells the program where to place some initial bricks before a simulation is run (similar to the pedicel introduced in Chapter 2).

C++ is the second approach's programming language of choice. Graphics and process management are accomplished by Ian Burleigh's VIGO library [3]. To optimize the routines of collision detection, a *loose octree* is dynamically built up and folded together during the simulation:

If an object moves in space and a collision with another object is supposed to initiate an event, the simulation has to query for intersection with all other objects in each step. If the space of the simulation in question is divided into small sections, collision queries only have to consider the particles that are in the same area as the agent. The mentioned octree automatically partitions space according to the number of objects in the simulation. The pruned search leads to a better performance (a profound introduction into collision detection and other computer graphics issues can be found in [1]).

<i>Simulation parameters</i>	
Spatial dimensions	$5 \times 10 \times 5$
Minimal distance of two particles	0.001
Block edge size	0.15
Agent sphere radius	0.05
Number of individuals per swarm	25
Simulated seconds	500
<i>Evolution parameters</i>	
Population size	20 swarms
Number $k$ of best genotypes	10
Crossover rate	0.4
<i>Mutation Rate</i>	
In general	0.2
Rules' actions	0.1
<i>Mutation Distance</i>	
Flocking parameters	0.05
Rules' conditions	0.2
<i>Alleles' Boundaries</i>	
Flocking parameters	$[0, 2]$
Rule vectors	The simulation's outmost points
Maximal number of rules	20
Maximal number of conditions	5

Table 3.7: Settings of the rule-based approach's phenotype simulation and evolution process

## 4 Evaluation of the two approaches

In the last two sections of the previous chapter, two different approaches to the evolution of a creative swarm are presented. Due to the very different claims of their actual swarm agent model and their particular evolutionary methods, this chapter presents some equally different results.

### 4.1 Strengths and weaknesses of the connectionist approach

Interactive evolution has too great a task to face, when hundreds of parameters need to be found. This truth does not hold in general. But if it takes a long time to overcome one search step and it is foreseeable that the number of intermediate states is immense, automatic search is the only option. Even after optimizing and parallelizing the simulation, the connectionist approach fails due to that reason.

Anyways, the scarce data that has been collected about the connectionist swarm, yields some interesting results, too. Smoothness and implicit dependencies of the construction elements for example.

#### 4.1.1 An example: Construction of maximum height

The first generation of a simulation run is shown in Figure 4.1. With a random initialization of the connectionist swarm's weight matrix, a colourful and diversified construction is produced. After a supervisor has guided the evolutionary process with the objective of building as high as possible for more than two hours, the population as seen in Figure 4.2 has been developed.

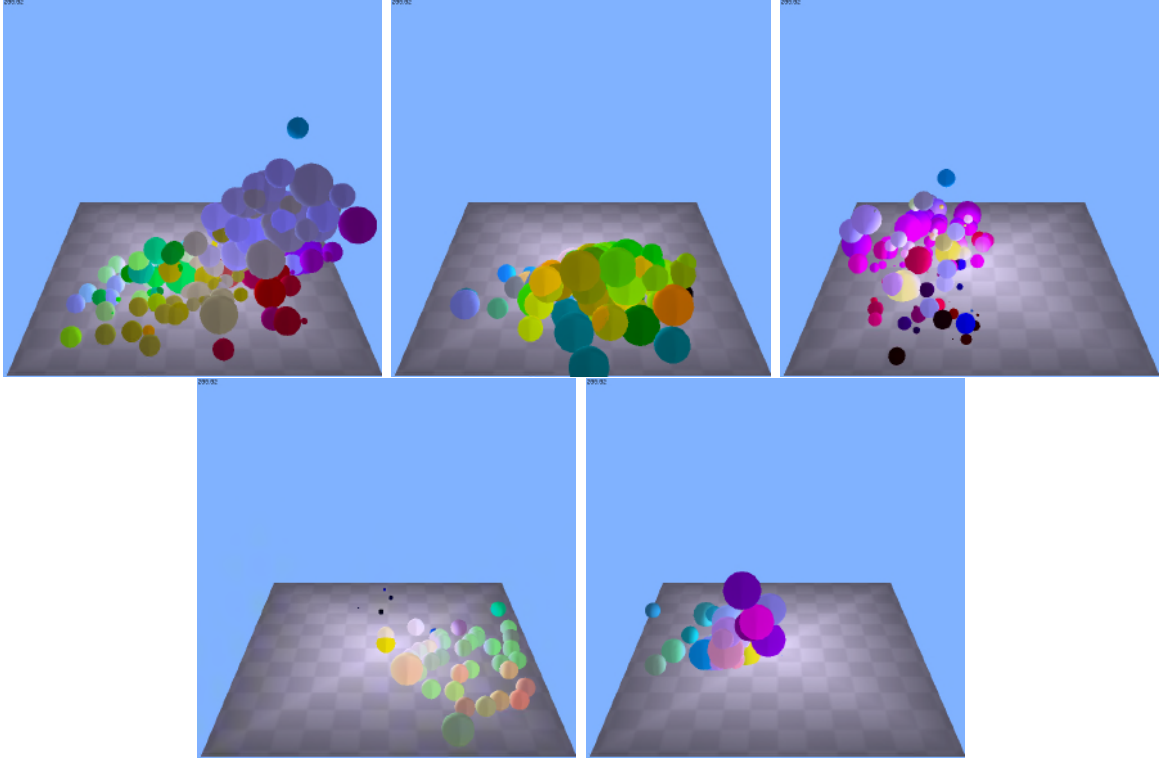


Figure 4.1: The construction phenotype of a randomly initialized connectionist swarm population.

A glimpse on several outcomes reveals that the inclusion of the agents' positions in the constructional network yields very one-sided structures in the full sense of the word. The simulation space is divided into four quadrants, the geometrical origin is the ground's center. It can easily be recognised that the majority of particles is concentrated within one of the four quadrants. This occurrence is induced by the agents' position vector that influences the swarm's construction. The location vector's components,  $x$ ,  $y$  and  $z$ , are connected to the perceptron that decides, whether to build a new particle or not. For each particle, the weighted sum of input values has been sufficiently great to conclude its construction. The images of Figure 4.3 clearly show the link between the agents' locations and the built particles.

Throughout all provided images the particles' attributes are connected to their positions. A concentric order of height, colour and size arises. These smooth transitions

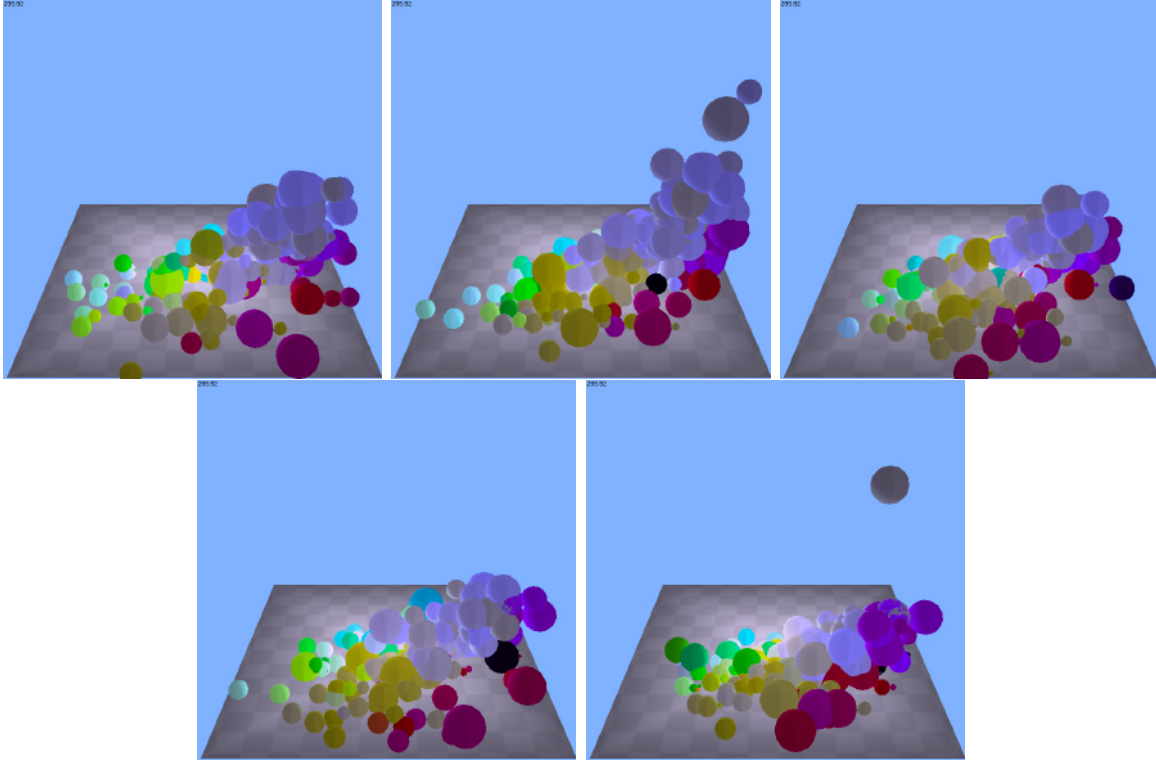


Figure 4.2: Fourteen generations later, the constructions of Figure 4.1 have evolved to this set of structures. The higher a structure was, the better was the swarm’s rating.

look nice. On the other hand this phenomenon makes clear, that the dependencies between the agents’ perception and actions overly bias the swarm’s construction.

#### 4.1.2 Fatal inefficiency

The most fatal encountered problem is the long computation time needed to produce the results of a swarm’s genotype and to evaluate them interactively. Henry Kwong (in [22]) has pointed out that:

- “Evolution may take longer, since humans judge solutions much slower than a machine in many cases.” and

- “For hard problems that require large populations and many generations, it is impractical for a human to perform the thousands of evaluations that would be necessary.”

Both cases apply in this approach.

Even though visualization is switched off and several machines compute a whole population in parallel, the computation of one generation takes a couple of minutes (depending on the number of particles in each simulation). Rating the results has to take place on one computer, which ends in another latency: The subsequent transfer of the results to one machine is done fast. However, displaying the phenotypes takes another long time.

This concomitant of the evolutionary process is even worse because the search space of weight matrices is immensely huge. Although a genetic algorithm and its derivatives usually produce good solutions to problems of low dimensionality, they tend to a poor performance if the genotype holds too many variables or the resulting genome depends on a very high precision [28]. Therefore the weight matrices need a lot of adjustment, if the swarm is meant to evince a specific construction behaviour. Assigning several minutes to each of the presumably many steps of the evolutionary search makes this approach a failure.

In total, processing one generation takes at least seven minutes: About two minutes for the phenotype’s computation, another two for visualization and, if done really quickly, three minutes to rate the outcomes. That makes approximately eight generations (of a population of only five individuals) in one hour. A hypothesis encompasses 315 numbers, each of them ranges from minus to plus five right from the start. With the assumption that whole number steps are sufficient to find good weight matrices, there are  $10^{315}$  distinct instances in the hypothesis space. The prospect of the objectives’ achievement can only be justified, if the supervisor’s rating radically prunes the search, the genetical operators work effectively together and one search step can be executed reasonably fast.

One might argue that the weight matrix (see Subsection 3.2.2) considers needlessly much input information. Despite the fact that it is unknown, which of the incoming data is irrelevant, discarding the consideration of ill-founded input would still end in an awfully<sup>1</sup> long procedure.

---

<sup>1</sup>Though uncommon in sciences, this term is used on purpose. It expresses the fact that too long latencies inconvenience the human supervisor.

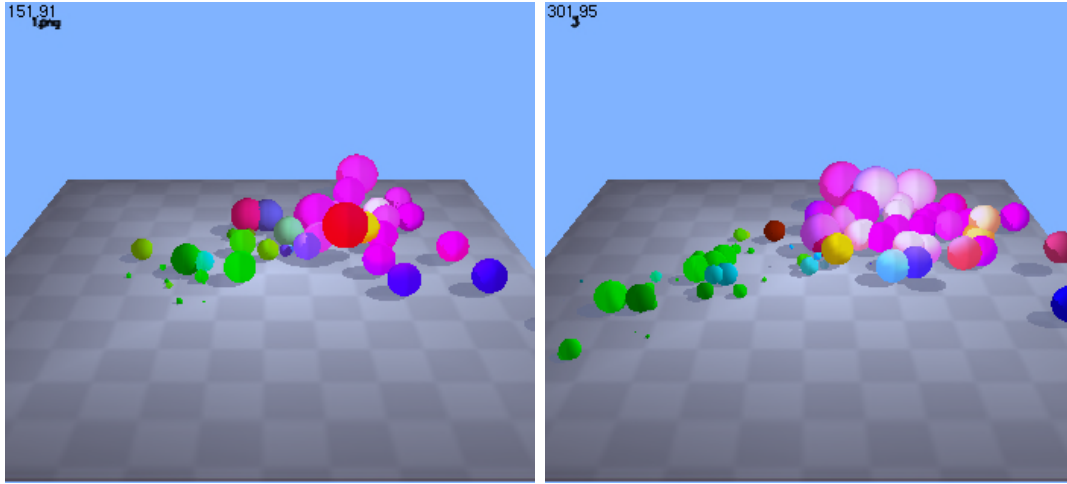


Figure 4.3: The location vector of the agent strongly influences the perceptron’s decision to build particles. The number of built particles varies according to the agents’ position.

### 4.1.3 Nice features of the connectionist approach

The simulation runs cannot refute the suitability of the connectionist swarm model in general. The smooth, colourful and diverse appearance of the created structures (especially in Figure 4.1) make the connectionist approach an aesthetical success (see Section 2.3). The direct connection between the created particles and such incoming data as the agent’s position do not seem to be adequate (the effects are stated in Subsection 4.1.1). However, though not very obvious, a tendency towards height is even visible in the pictures of the “height objective’s” 15th generation (Figure 4.2).

One may conclude that particles of different sizes and colours are a good basis for an aesthetically appealing appearance. The presented connectionist agent model is capable of producing the necessary continuous outcome values for that purpose.

## 4.2 Evaluation of the rule-based approach

After the presentation of some interesting structures built by rule-based swarms, a case-study is adduced. The case-study comprises the swarm’s evolutionary development, attributes of its created structure and characteristics of its flocking behaviour. Finally a

few examples show how an already evolved swarm is interactively guided, to approximate derivations of its original fitness structure.

Mapping three-dimensional contents onto normal 2D images is a rather difficult process. The upcoming pictures are taken in such a way that as much as possible of the important information is cognizable. At the same time this works against the attempt to provide uniform pictures.

### 4.2.1 Exemplary structures

By means of the techniques presented in Section 3.3 many rule-based swarms have been evolved that create interesting structures (see Figures 4.4, 4.5, 4.6, 4.7 , 4.8, 4.9). Table 4.1 shows the swarms' fitnesses, the generation in which they occurred and the underlying structures that have guided the evolutionary search. Except for the swarms of Figures 4.5 and 4.7 which have both a set of 18 rules, all the presented swarms have 10 constructional rules and make use of maximally 4 to 5 conditions.

Although the construction in Figure 4.4 seems to be absolutely conforming to its underlying "fitness structure" (first image of Table 4.1), it has only acquired a puny fitness. Appearances are deceiving, the evolved swarm is merely a by-product of the evolutionary search. In fact, the swarm builds the desired straight line exactly ninety degrees into the wrong direction. However, it has been discovered by chance and satisfies the experiment's expectations.



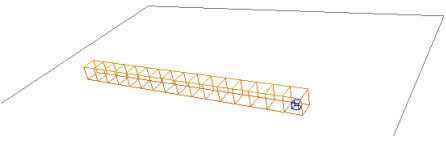
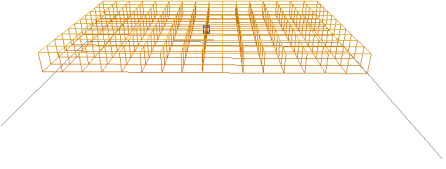
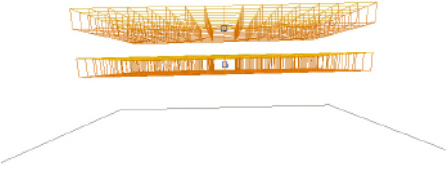
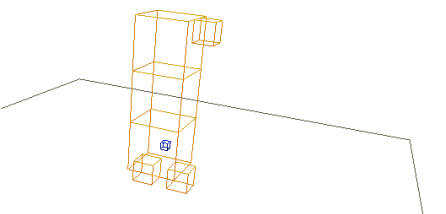
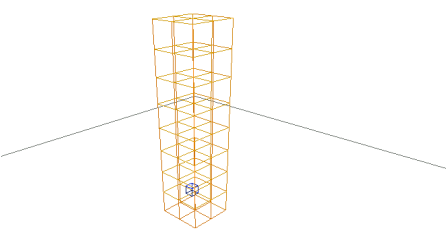
<i>Fitness Measuring Structures</i>	<i>Referenced Result(s)</i>	<i>g</i>	<i>f</i>
	Parallel Lines, Figure 4.4	15	0.0064
	Fan-like Shape, Figure 4.5	506	0.0058
	Two-level Flats, Figure 4.7	314	0.0122
	Unsymmetric Structure, Figure 4.8	7	0.1305
	Bush Structure, Figure 4.6 Organic Structure, Figure 4.9	1143 35	0.0066 0.1970

Table 4.1: Structures that have guided the evolutionary search are coloured in red, seed blocks in blue. The corresponding results are referenced on the right side. Furthermore it is stated in which generation  $g$  the result has occurred and what fitness  $f$  it has achieved.

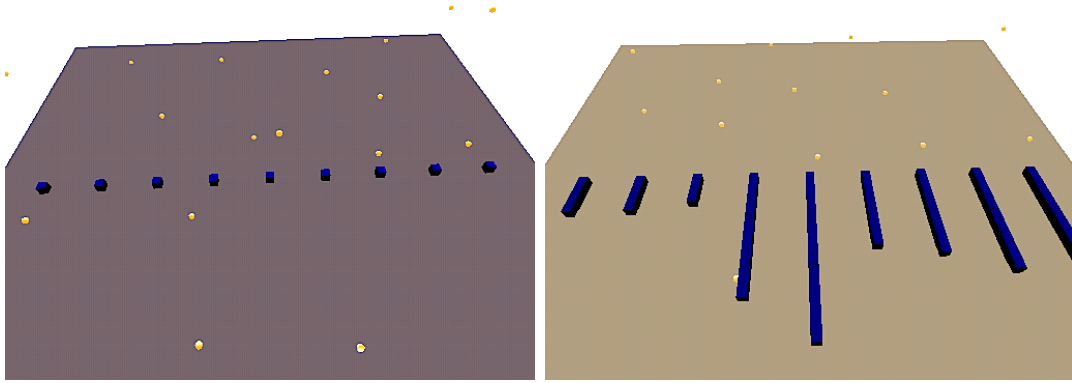


Figure 4.4: On the left there is a line of seed blocks that trigger the swarm’s building process. The right image shows the swarm’s constructional result. The almost equal distribution of the swarmettes along with their low flight supports their constructional efforts.

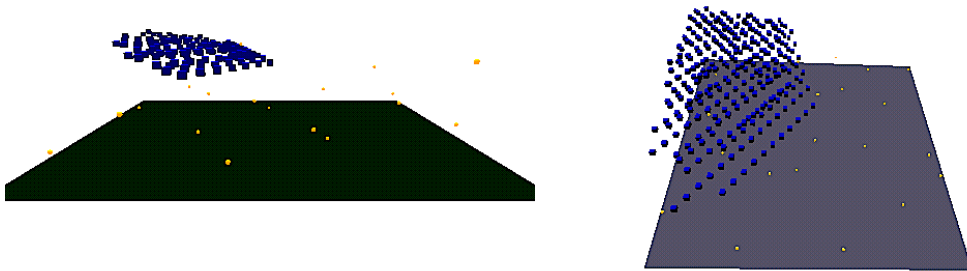


Figure 4.5: The fan-like construction has been achieved by starting from a single seed block in the simulation world’s center, a small distance from the ground. The image on the left side shows an earlier stage of the construction’s development.

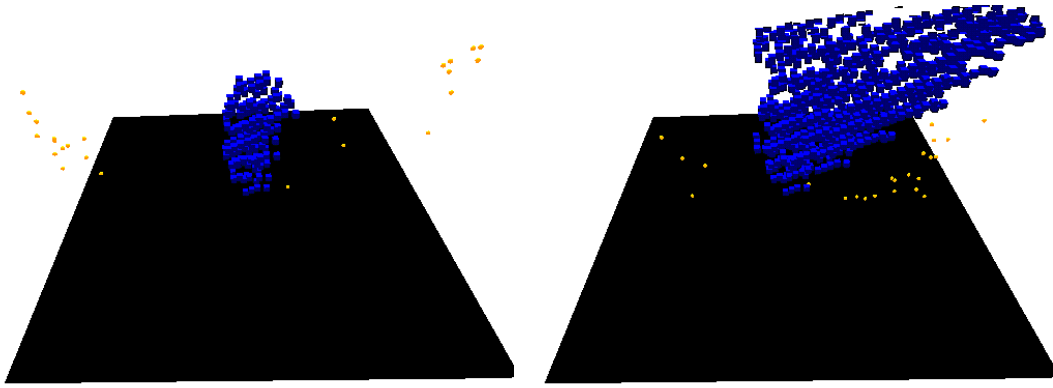


Figure 4.6: The swarm is divided into two flocks at an early stage. Both flocks loop back and forth from their sides to the construction. The many holes in the building make it look like a bush.

#### 4.2.2 A case-study: Approximation of a tower

The tower “fitness structure” displayed in the last image of Table 4.1 has also been basis of the development of the swarm which is examined in this subsection. In general the swarm in question has similar attributes as the ones presented in Subsection 4.2.1: Its constructional behaviour is determined by a set of ten rules, whereas the maximum amount of conditions of these rules is five. It took 143 generations for the considered swarm to occur and it achieves a fitness of 0.1926.

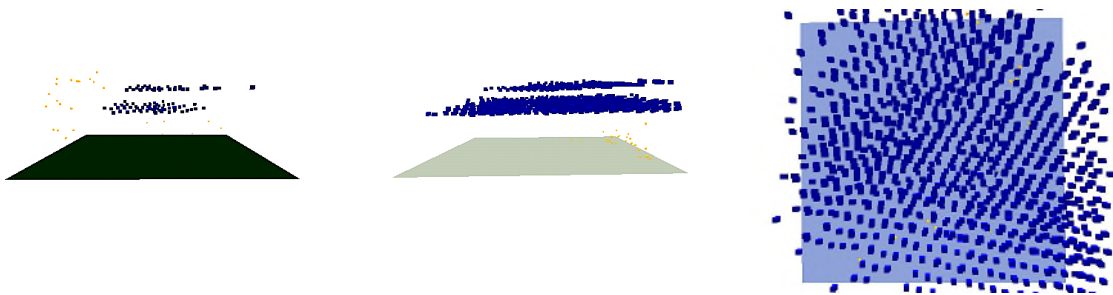


Figure 4.7: Two-level flats: Developed from two seed blocks at the corresponding heights. The third image shows the construction from above.

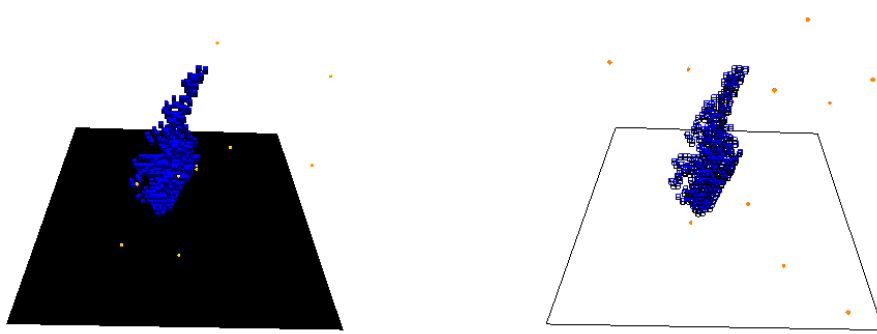


Figure 4.8: The more interesting shapes are unsymmetrical like this one. The right picture enhances the view on the swarm's flocking behaviour.

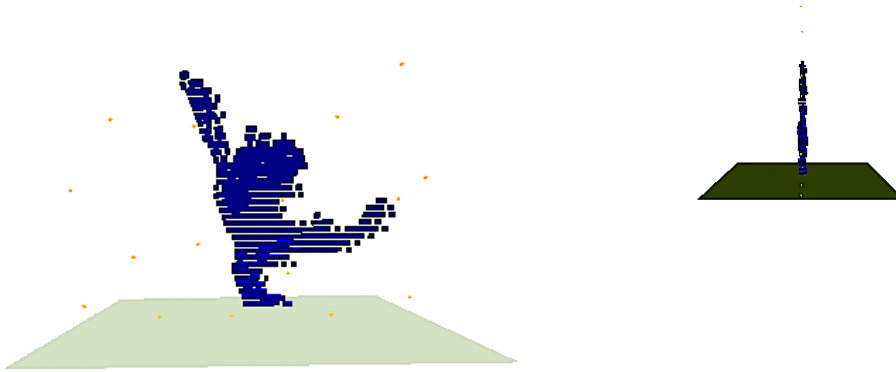


Figure 4.9: An excellent example of an asymmetric, organic looking shape. But it has only two dimensions: The right picture shows the structure from the side, no swarmette ever leaves the plane.

But the swarm's overall behaviour is exceptional. First it builds up a tower which is very close to the given structure in no time (Figure 4.10), then it is fleeing from the world's center to avoid further construction which might lead to fitness penalty (Figure 4.11). The whole construction procedure takes approximately 200 simulated seconds, but the biggest part is already built after 80 seconds have passed. At the simulation's start and as long as it takes to create the tower, the swarm stays around the center.

Afterwards it is divided into three to four flocks which head as far away from the center as possible, lingering in the simulation world's corners.

In Chapter 2 compactness, structure and coordination of a swarm's creation are explained. With respect to these keywords the present construction can be characterised as follows:

**Compactness:** The particles are built close to each other, therefore the construction is compact.

**Structure:** Although an iterative construction process can be observed (Figure 4.10), patterns or modules of bricks are not apparent. However, the fact that the bricks are built on top of each other to gain height and next to each other to approximate the given shape's breadth is sufficient to outline a structure (after the first description of a structure in Section 2.1 on page 8).

**Coordination:** The single construction steps of Figure 4.10 clearly show that it occurs that several agents build at different locations without jeopardizing the total construction. Therefore one may say that the construction is coordinated.

Another question arises when looking at the single steps of Figure 4.10: *How does the swarm change its flocking behaviour?* To answer this question the agents' genotype has to be investigated (the detailed genotype can be found in the Appendix). It consists of the following rules:

- An unconditional rule that recants the declaration of the swarm's center
- Five rules adding a new particle
- Two rules destroying a block
- Two rules rearranging the swarm's center

The succession of the rules plays an enormous role. Exempting the swarm from its center is only put into practice, if no subsequent rule defines the swarm's center anew. Since the rules that redefine the swarm's center ask for certain structural configurations around the agent to come into effect, the swarm is focused on the building as long as these conditions are fulfilled. Through steady alteration of the built structure it might occur

that these conditions are not satisfied for agents at a certain location. As a consequence the swarm’s intrinsic flocking behaviour could come into action urging the swarm to the simulated world’s corners. This reasoning conforms to the seen phenomenon and is supported by the swarm’s genotype.

### 4.2.3 Guiding the search with diversified 3D objects

The perfect rule-based tower building swarm (presented in Subsection 4.2.2) motivates to make use of its properties in the following way:

- Use the elite (the ten best genotypes) of the “height approximation” experiment as starting point for a new evolutionary run (a combination of elitism and a multi-start hillclimber strategy, presented in 2.3)
- Alteration of the 3D structure which has been successfully approximated

Three different ideas for the tower shape’s change are discussed:

1. The simple extension of the tower’s height
2. Breaking open the close and compact structure of the tower
3. Building some stairs on top

The new structures are shown in Figure 4.12 (in the same succession). Their solid appearances result from the facts that the shape’s overall structure can be recognised more easily this way and covering the seed block may be done because it is at the same position as in the example of Subsection 4.2.2.

It took 184 generations until the tower building swarm (introduced in Subsection 4.2.2) has adopted to the “fitness structure’s” elongated shape. In general the same course of events takes place as seen in Figures 4.10 and 4.11: Now the agents build a somewhat higher tower before they flee from the world’s center. Although the swarm does well, the task’s similarity to the approximation of the smaller tower diminishes its success.

More interesting results of the tower extension experiment are displayed in Figure 4.13. The images represent the results of two separate evolutionary runs, both starting with the genotype of the tower building swarm as discussed above. It is obvious that

there is a general tendency of the swarms to fulfil the new requirements. The extension of the original tower's shape gives a strong impetus to build higher. It may be stated that the evolutionary process considers the shape's variance and thereby generalises beyond the new given structure.

The second extension of the tower example, that is the branching crown on top of the tower, has been approximated within 1341 generations, Figure 4.14 presents the outcome.

Figure 4.15 shows the approximation of the third shape variation. It seems to be hard to cope with the stairs on top of the tower. However, after already two generations the hillclimber strategy yields a genotype with a fitness of 0.2087 which produces a structure bent into direction of the stairs. A simulation run with no specific starting point has not been able to achieve equally good results within 1000 generations. Its best result has occurred in generation 925. The swarm in question has developed a slight tendency of building towards the stairs (overall fitness: 0.0401). The "tower with stairs" experiment combines two basically different structures, one straight upwards and the other one diagonally upwards. Since the genotype of the latter swarm is trained to approximate the tower shape and the stairs structure at the same time, the constructional tasks might be overextending.

In Section 4.1 the flaws of interactive evaluation are explained. In the presented connectionist approach (Section 3.2) the combination of two factors have led to failure: Too many parameters to optimize and too slow computation and visualization of the phenotypes. The examples in this subsection are guided by an underlying 3D structure and an external supervisor that changes the evolution process' objective. In this manner a stepwise development from "vague notions" of interesting or aesthetical structures to corresponding fitness shapes can take place.

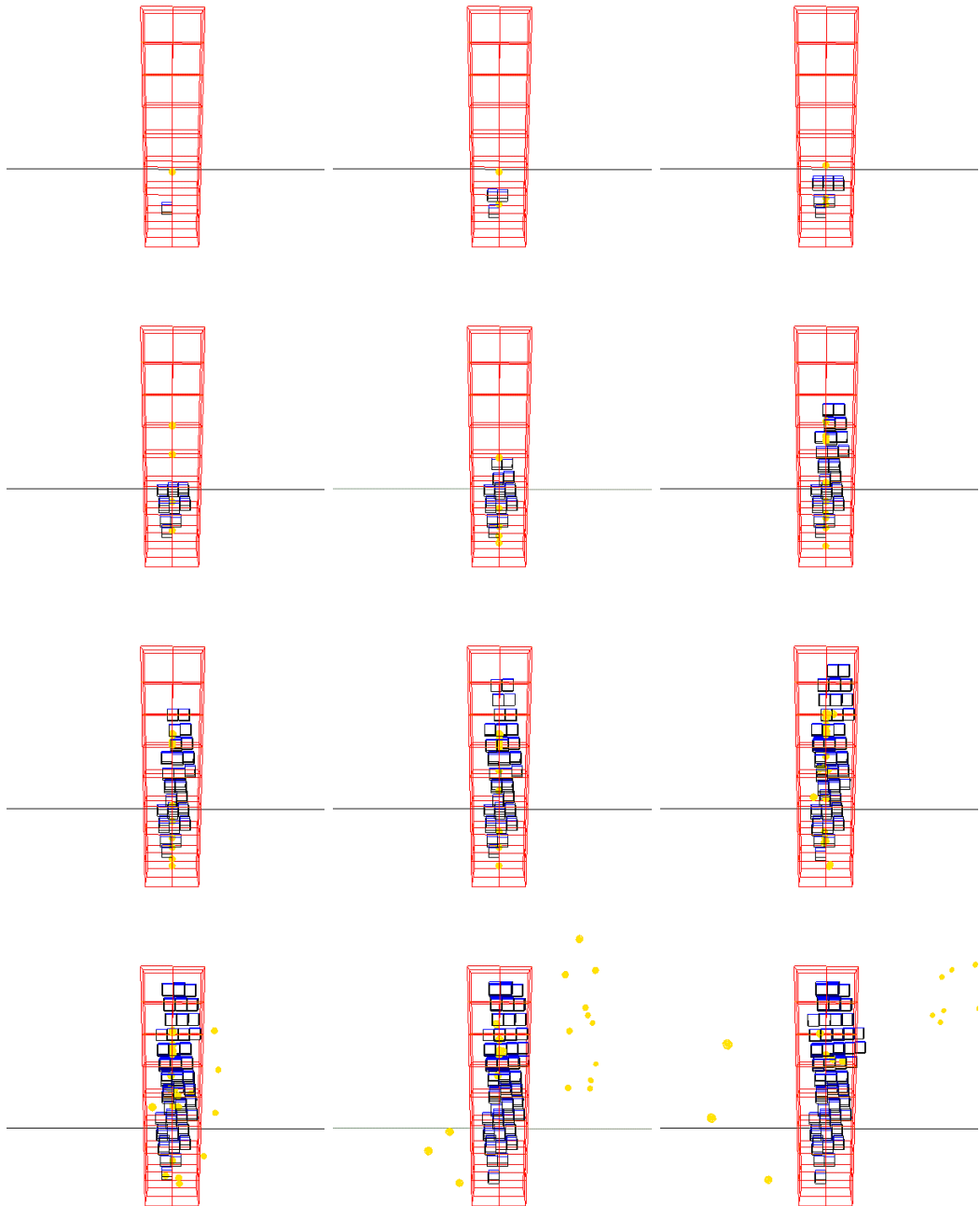


Figure 4.10: Intermediate states of the building process of a tower. The orange “fitness structure” elucidates the excellence of the swarm’s approximation.



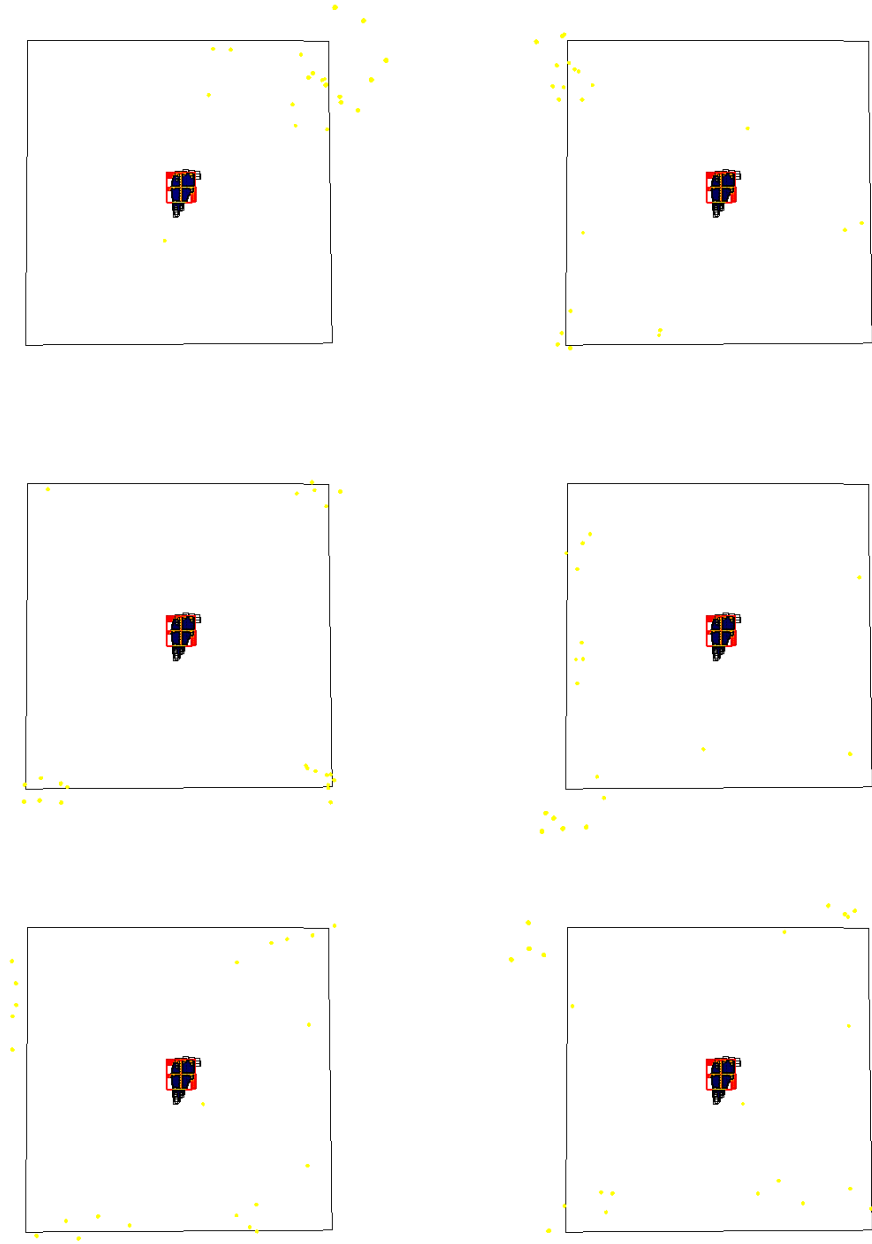


Figure 4.11: The swarm's flocking behaviour after it has successfully approximated the given tower structure. If the swarm gets into contact with the central building it might be triggered to extend the construction. Since particles outside the given structure is punished, the swarm is better off hiding in the world's corners.

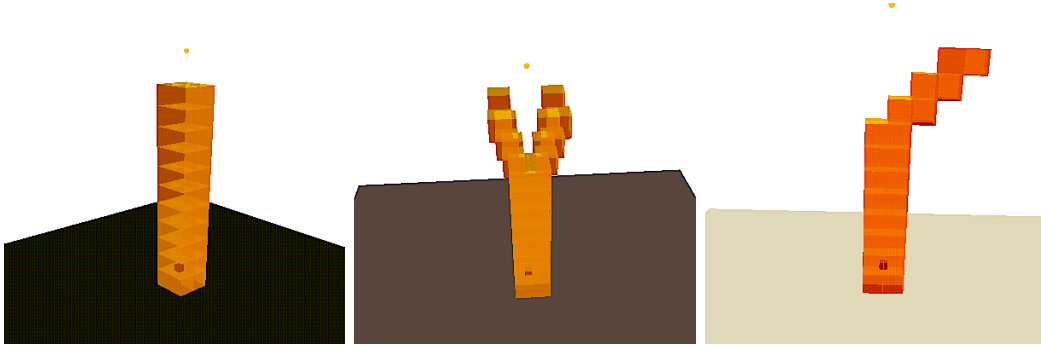


Figure 4.12: Extended versions of the original guiding structure (seen in the last image of Table 4.1): An extension of the tower's height, a branching "crown" and stairs on top of the tower.

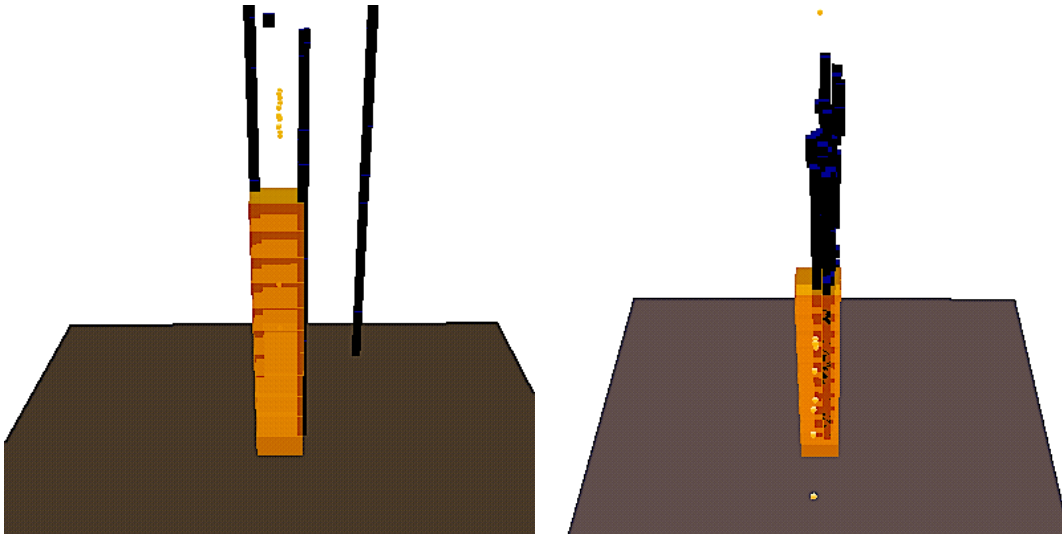


Figure 4.13: 1. Pillars arise after 1000 generations 2. Extension of the original "fitness structure" results in endless efforts to gain height (644. generation) Both images clearly show the vertical line flight formation that contributes to the swarm's construction

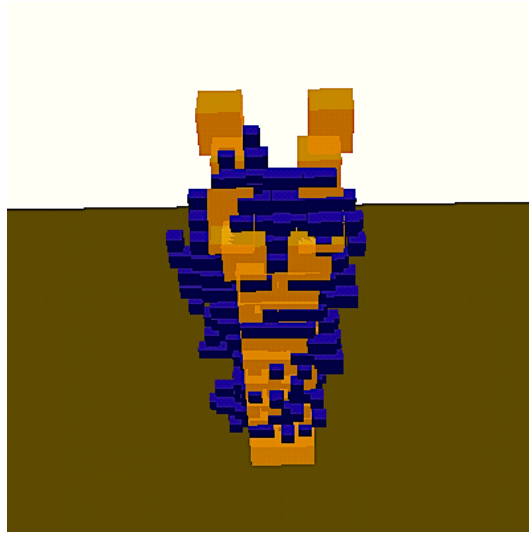


Figure 4.14: After 1341 generations the original tower building swarm has adjusted to the new challenge: Now it contributes to the branching structure by extending its construction's diameter with growing height.

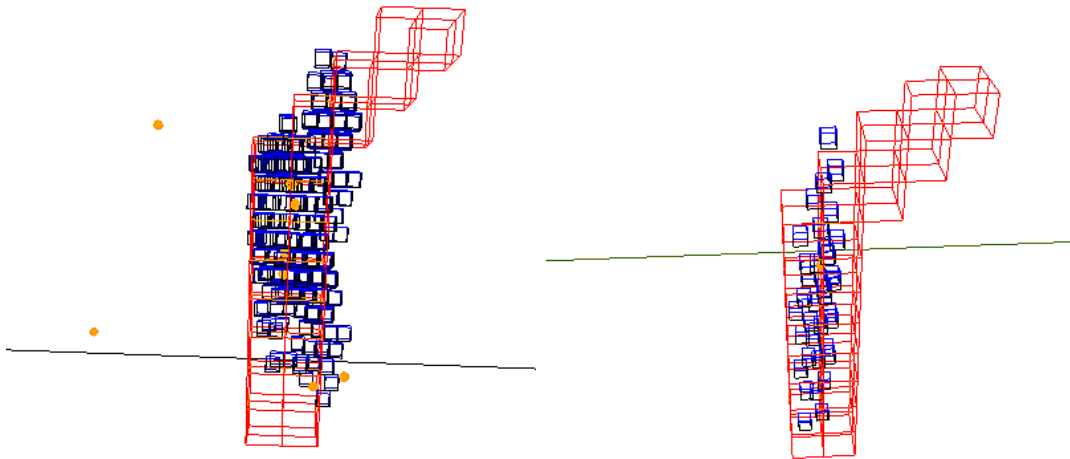


Figure 4.15: The first image shows the result of an evolutionary run with an optimized tower building swarm as basis. On the right side an outcome without a specific initialization is displayed (925. generation).

## 5 Summary and future work

Based on inspiration from insects' construction abilities and from the successful application of interactive evolution in cases where creative development is the objective, a swarm model and an appropriate evolutionary system have been implemented. Although the connectionist agent model was too complex to be evolved interactively, a positive discovery has been made: The connection between input information and the agents' actions in combination with the computation of continuous values leads to very smooth, colourful and diverse results. There might be applications, especially in the field of the fine arts, where such properties of an agent system are desired. Interactive art installations have not been able to gain a huge market share so far, but as offers of virtual reality gear become more and more attractive, there lies a great potential for aesthetical 3D applications. Still, the results of the first approach can neither affirm nor refute the main idea of the connectionist agent model.

The disappointment about the first approach's impasses has led to fundamental conceptual changes in the second attempt. The agent model is much simpler. For a start the agents only consider the surrounding configuration to determine their constructional actions. The lattice swarms, first presented by Bonabeau et al. [12] and later investigated by Pilat [29], that react on qualitative stigmergic signals, have served as a most promising model. The lattice swarm has been transferred into a continuous 3D space and the original random movement in space has been substituted by Reynold's flocking model [30]. The latter novelty makes way for the parallel evolution of a swarm's construction behaviour and its flocking parameters. A new actuator of the swarmettes has been introduced, too. Equipped with the functionality to actively change the swarm's center, the agents have the means to coordinate their work. This functionality also links the swarm's constructional with its flocking behaviour.

If an agent is supposed to consider qualitative stigmergical information a continuous 3D world, a new test for the surrounding structural configuration is required, therefore a proper solution is suggested in Figure 3.6. The most important extension is to let the swarm orientate itself towards a given three-dimensional structure (technically this is done by considering a 3D shape, the computation of the difference between the given and the built structure, see Algorithm 3.1 and an appropriate fitness function, see Equation 3.14).

In Chapter 4 some interesting structures of rule-based swarms are shown. The first examples are discovered by providing the search with several 3D structures (Table 4.1). In addition the supervisor can keep up with guiding the process: An already developed swarm can be bred to approximate an altered shape. Judging the swarm's fitness is conceded to the evolutionary system and the supervisor's guidance takes place by his/her supply of refined structures.

A huge amount of decisions has been made to develop a complete swarm model in tandem with an evolutionary system. Though many interesting structures have occurred, a lot of improvement might be achieved by a comparative development of the genetic operators in Section 3.3.4. Among others, the agent's genotype is limited to maximally five conditions per rule. So far artificial evolution has originated mostly swarms that obtain rules with maximally four to five conditions which clearly exhausts the given limit of conditions. Therefore the effect of an increased number of allowed conditions should be analysed.

Each of the presented swarms has one genotype which holds for all its individuals. It has been shown that the presented swarm model works in general and yields some interesting results. However, more complex structures might arise by assigning one genotype to each swarm agent. Since a whole set of distinct swarms is already found, it would be interesting to merge some of their individuals to let them create totally new structures. This new perspective enforces the attempt to evolve swarms consisting of "real" individuals.

# A Appendix, DNA of a Tower Building Swarm

The Genotype of a Rule-based Tower Building Swarm

```
<DNA>
<SWARMPARAMETERS>
<ALIGNMENT> 0.308151
<COHESION> 0.181587
<SEPARATION> 0.0599679
<RANDOM> 0
<GROUND> 0.166916
<CENTER> 0.13821
</END_SWARMPARAMETERS>
<RULE>
<ACTION> 2
<ACTIONPARAMETER> 0.255089 0.762202 0.554074
</END_RULE>
<RULE>
<ACTION> 5
<ACTIONPARAMETER> -0.712361 0.299528 0.0635564
</END_RULE>
<RULE>
<CONDITION> -0.14244 0.578193 -0.184357
<ACTION> 1
<ACTIONPARAMETER> 1.05214 0.745683 -1.7563
```

```
</END_RULE>
<RULE>
<CONDITION> -0.05616 0.0895128 -1.64249
<CONDITION> -0.793849 0.582498 0.605538
<CONDITION> 0.640603 0.155417 -0.964139
<CONDITION> 0.475184 0.191081 -0.227879
<ACTION> 1
<ACTIONPARAMETER> -0.79103 1.41344 1.0936
</END_RULE>
<RULE>
<CONDITION> -0.464115 1.25665 0.0482793
<CONDITION> -0.272851 0.497835 0.419289
<ACTION> 3
<ACTIONPARAMETER> -0.0553006 0.288848 -0.78648
</END_RULE>
<RULE>
<ACTION> 3
<ACTIONPARAMETER> -0.0233629 0.190972 -0.0661236
</END_RULE>
<RULE>
<CONDITION> -0.383928 0.00313065 -0.233784
<ACTION> 3
<ACTIONPARAMETER> -0.334766 0.37158 0.518885
</END_RULE>
<RULE>
<ACTION> 3
<ACTIONPARAMETER> 0 0 0.151
</END_RULE>
<RULE>
<ACTION> 3
<ACTIONPARAMETER> 0.133642 0.194942 -0.0436256
</END_RULE>
```

```

<RULE>
<CONDITION> 0.0106957 0.165715 -0.139416
<ACTION> 4
<ACTIONPARAMETER> -0.634567 0.944219 0.823604
</END_RULE>
</END_DNA>

```

<i>Action Number</i>	<i>Description</i>
0	Set center to itself
1	Set center to a specific location
2	Recant swarm center
3	Create new particle
4	Destroy the collision particle
5	Destroy a remote particle

Table A.1: The rule-based agent's actuators corresponding to the genotype numbers.



# Bibliography

- [1] T. Akenine-Möeller and E. Haines. *Real-time Rendering, Second Edition*. AK Peters, Natick, Massachusetts, 2002.
- [2] T. Bräunl. *Research Relevance of Mobile Robot Competitions*. In *Robotics & Automation Magazine, IEEE*, Vol. 6, Issue 4, pages 32-37. IEEE Press, Dec 1999.
- [3] I. Burleigh. *VIGO*. <http://pages.cpsc.ucalgary.ca/~burleigh>, 2004.
- [4] P. Callahan. *Wonders of Math - The Game of Life*. The World of Math Online Math.com, <http://www.math.com/students/wonders/life/life.html>, 2005.
- [5] R. Dawkins. *The Blind Watchmaker*. Longman Scientific and Technical, Harlow, 1987.
- [6] K. Dejong. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD Thesis, University of Michigan, 1975.
- [7] J.-L. Deneubourg. *Application de l'ordre par Fluctuations à la Description de Certaines étapes de la Construction du nid chez les Termites*. Insect. Soc. 24, pages 117-130, 1977.
- [8] M. d'Inverno and M. Luck. *Understanding Agent Systems*. Springer-Verlag, Berlin, Heidelberg, 2001.
- [9] I. Ravn (ed). *Chaos, Quarks und schwarze Löcher: Das ABC der neuen Wissenschaften*. Verlag Antje Kunstmann, München, 1995.
- [10] J. Crowther (ed). *Oxford Advanced Learner's Dictionary of Current English, Fifth Edition*. Oxford University Press, 1995.

- [11] Bentley et al. *New Trends in Evolutionary Computation. In Proceedings of CEC 2001, the Congress on Evolutionary Computation, Seoul, Korea.* pp. 162-169. RN/00/68. IEEE Press, May 27-30, 2001.
- [12] E. Bonabeau et al. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, Oxford, 1999.
- [13] I. Burleigh et al. *DNA in Action! A 3D Swarm-based Model of a Gene Regulatory System. In: Proceedings of the First Australian Conference on Artificial Life. Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, 2003.
- [14] Wright et al. *Implicit Parallelism. In Proceedings of GECCO 2003 (Lecture Notes in Computer Science, vols 2723-2724), E. Cantu-Paz (eds)*. Springer-Verlag, Berlin, Heidelberg, 2003.
- [15] G.W. Flake. *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaption*. A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England, 1999.
- [16] D. Fogel. *Blondie 24: Playing At The Edge of AI*. Morgan Kaufmann, 1999.
- [17] J. Holland. *Adaption in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan, 1975.
- [18] C. Jacob. *Illustrating Evolutionary Computation with Mathematica*. Morgan Kaufmann, San Francisco, 2001.
- [19] C. Jacob. *IEC web site*. <http://pages.cpsc.ucalgary.ca/~jacob/IEC>, 2004.
- [20] E. Klavins. Toward the control of self-assembling systems. In A. Bicchi, H. Christensen, and D. Prattichizzo, editors, *Control Problems in Robotics*, volume 4 of *Springer Tracts in Advanced Robotics*, pages 153–168. Springer, 2003.
- [21] J. Klein. *Breve Website*. <http://www.spiderland.org/breve>, 2004.
- [22] H. Kwong. *Evolutionary Design of Implicit Surfaces and Swarm Dynamics Master Thesis*. University of Calgary, 2003.

- [23] H. Kwong and C. Jacob. Evolutionary exploration of dynamic swarm behaviour. In *Congress on Evolutionary Computation*, Canberra, Australia, 2003.
- [24] C.P. Lieckfeld. *Ästhetik. In Bionik: Natur als Vorbild*. Pro Futura Verlag GmbH, München, 1993.
- [25] S. Mandl and H. Stoyan. *Evolution of Agent Coordination in an Asynchronous Version of the Predator-Prey Pursuit Game*. In Lindemann et. al., editor, *MATES 2004 (Multiagent System Technologies Erfurt 29.9.2004 - 30.9.2004)*, Bd. LNAI 3187, p. 47-57. Springer-Verlag, Berlin, Heidelberg, 2004.
- [26] M. Minsky. *The Society of Mind*. Simon and Schuster, Inc., New York, 1988.
- [27] T. M. Mitchell. *Machine Learning*. McGraw Hill, Boston, Massachusettes, 1997.
- [28] I.C. Parmee. *Evolutionary and Adaptive Computing in Engineering Design*. Springer-Verlag, London, Berlin, Heidelberg, 2001.
- [29] M.L. Pilat. *Wasp-Inspired Construction Algorithms*. <http://www.pilat.org/>, University of Calgary, May 1, 2004.
- [30] C.W. Reynolds. *Flocks, Herds, and Schools: A Distributed Behavioral Model*. In Computer Graphics, 21(4) (SIGGRAPH '87 Conference Proceedings) pages 25-34, 1987.
- [31] K. Richter and J.-M. Rost. *Komplexe Systeme*. Fischer Taschenbuch Verlag, Frankfurt am Main, 2002.
- [32] R. Rojas. *Neural Networks: A Systematic Introduction*. Springer-Verlag Berlin Heidelberg, 1996.
- [33] P.T. Saunders and M.W. Ho. *Thermodynamics and Complex Systems*. In C.W. Kilmister, editor, *Disequilibrium and Self-Organisation*, pages 243-253. D. Reidel Publishing Company, Dordrecht, Holland, 1986.
- [34] K. Sims. *Artificial Evolution for Computer Graphics*. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 319-328. ACM Press, New York, USA, 1991.

- [35] L. Spector and J. Klein. *Evolutionary Dynamics Discovered via Visualization in the Breve Simulation Environment*. In *Artificial Life VIII*, Reading, MA. Addison-Wesley, 2002.
- [36] D. Thomas. *Aesthetic Selection of Developmental Art Forms*. In *Artificial Life VIII. The 8th International Conference on the Simulation and Synthesis of Living Systems, Sydney, Australia, pages 157-163*. MIT Press, Cambridge, December 9-15, 2002.
- [37] Trolltech. *Qt is a C++ toolkit for multiplatform GUI & application development*. <http://www.trolltech.com/qt/>, 2003.
- [38] E.W. Weisstein. *Langton's Ant*. From MathWorld—A Wolfram Web Resource. <http://mathworld.wolfram.com/LangtonsAnt.html>, 1999.
- [39] T. Whitelaw. *Breeding Aesthetic Objects: Art and Artificial Evolution*. In *Creative evolutionary systems, Section: Evolutionary creativity, pages 129-145*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.
- [40] The free encyclopedia Wikipedia. *Aesthetics in Art. Aesthetics in Architecture*. <http://en.wikipedia.org/wiki/Aesthetics>, 7 Jan 2005.