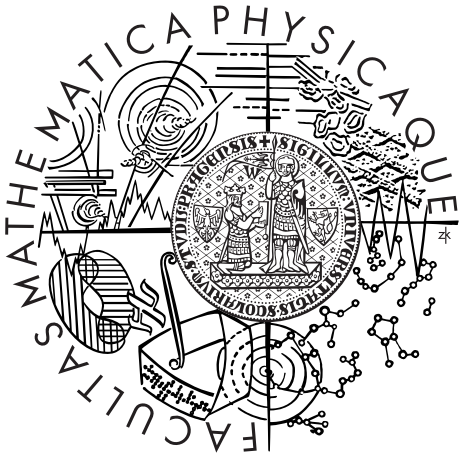
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Milan Rybář

Inspiration-triggered search: Towards higher complexities by mimicking creative processes

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis: Jun.-Prof. Dr. Heiko Hamann
Study programme: Informatics
Specialization: Theoretical Computer Science

Prague 2015

I would like to thank my supervisor Jun.-Prof. Dr. Heiko Hamann for giving me such a life-time opportunity while I was an exchange student at the University of Paderborn. I am grateful for his guidance, strong interest in the topic, and his friendly approach. Additionally, I would like to thank Charles University in Prague, University of Paderborn, and Erasmus+ programme from the European union that I could study two unforgettable semesters abroad at the University of Paderborn in Germany.

I would like to thank Petr Fanta for the shared struggle while working on ours bachelor and master theses. I would also like to thank Abhishek Ananthram, Frank Kluthe, and Zahra Nouri for the shared moments while working on this thesis in a computer pool in Zukunftsmeile 1.

I would also like to express my gratitude to my family who always supported me during my studies.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague on 31st July 2015 Milan Rybář

Název práce: Inspiration-triggered search: Za vyššími složitostmi napodobováním tvůrčích procesů

Autor: Milan Rybář

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: Jun.-Prof. Dr. Heiko Hamann, Department of Computer Science, University of Paderborn, Německo

Abstrakt: Jeden z hlavních problémů stochastických optimalizačních metod ze strojového učení je uvíznutí v lokálních optimech. Cílem této práce je vytvoření optimalizační metody inspirované uživateli webové služby Picbreeder, ve které mohou společně vyvíjet obrázky pomocí umělé evoluce. Hlavní myšlenkou je, že jejich chování představuje tvůrčí procesy. Představujeme metodu nazvanou inspiration-triggered search, která napodobuje zmíněné procesy a využívá k tomu libovolnou optimalizační techniku. Vyhledávání neobsahuje pevně daný cíl, místo toho je schopno samo si s určitými omezeními definovat vlastní cíle. Cílem optimalizace je vytvoření komplexních výtvorů, které nemohou být nalezeny hladovou a přímou optimalizací. Navržená metoda je otestována v doméně obrázků, kde je cílem nalezení komplexních a esteticky příjemných obrázků pro člověka, a porovnána s přímou optimalizací.

Klíčová slova: evolučně výpočetní techniky, předčasná konvergence, vyhledávání bez cíle, tvůrčí proces, evoluční umění

Title: Inspiration-triggered search: Towards higher complexities by mimicking creative processes

Author: Milan Rybář

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Jun.-Prof. Dr. Heiko Hamann, Department of Computer Science, University of Paderborn, Germany

Abstract: The trap of local optima is one of the main challenges of stochastic optimization methods from machine learning. The aim of this thesis is to develop an optimization algorithm that is inspired by users interacting with Picbreeder, which is an online service that allows users to collaboratively evolve images via an artificial evolution. The idea is that their behaviours depict creative processes. We propose a general framework on the top of a common optimization technique called inspiration-triggered search, which mimics these processes. Instead of a fixed objective function the search algorithm is free to change the objective within certain constraints. The overall optimization task is to generate complex artefacts that cannot be generated by a greedy and direct optimization approach. The proposed method is tested in the domain of images, that is to find complex and aesthetically pleasant images for humans, and compared with the direct optimization.

Keywords: evolutionary computation, premature convergence, non-objective search, creative process, evolutionary art

Contents

Introduction

The trap of local optima is one of the main challenges of stochastic optimization methods from machine learning [2]. They search an optimal solution representing a goal in a particular domain guided by an objective function. The typical objective function rewards getting closer to the goal. Nevertheless, the direct path to the goal may contain local optima. In order to get to the goal, the greedy search cannot only straightforwardly follow the deceptive objective function. It must make detours, which is moving away from the goal. Otherwise, the search may prematurely convergence in suboptimal solutions. One work [20] proposes a possible explanation of this problem. The objective function serves two different purposes: it defines the goal and guides the search. Mixing up these two purposes works well when finding good solutions does not impose large detours that are not directly identifiable with the objective function. However, recent experiments showed several tasks in which this kind of objective-based search was especially ineffective [45, 104].

As a rule of thumb, the more complex a task the more likely the search can be deceived by local optima. Many techniques have been proposed to extend existing methods and scale them to more complex tasks. Examples of such general techniques are simulated annealing, tabu search, guided local search, iterated local search, and scatter search. Within the field of evolutionary computation [22], examples of such approaches are techniques based on behavioural diversity [59]. Recent experiments have demonstrated that guiding the search with an objective function is not the only possibility. For instance, novelty search [43, 45] is not driven towards any particular goal. Instead it only searches for novel behaviours. The authors argue that sometimes not looking for the goal in this way leads to finding the goal more quickly and consistently [93].

Furthermore, an objective function for complex tasks is difficult to craft. For instance, let us imagine, we want to evolve a neural network that would allow a mobile robot to avoid obstacles [65]. A straightforward objective function of minimizing the number of collisions will probably be disappointing. The robot, that does not move at all, receives the maximum score. If the objective function is improved to force the robot to move, the robot will probably move in a circle instead of exploring the environment. As illustrated by this example, a long refinement process is often required before obtaining an objective function that unambiguously reflects the target behaviours. This is often achieved by an incremental approach with a sequence of objective functions, each chosen to build upon the previous [23, 27]. Yet such incremental training is difficult and requires intimate domain knowledge and careful oversight.

For a broader view, we can look into nature. Fossils provide many examples that would have been hard to find using objective-based search. Many traits of animals and plants have been co-opted for a purpose for which they were not initially selected [29]. Classic examples are bird feathers, which may initially have evolved for temperature regulation, and vertebrate bones, which may have been selected to store phosphates. Similarly, the history of science and technology has many examples of serendipitous discoveries [73, 31]. For instance, the effects of penicillin on microbes were first observed in a failed experiment about lysozyme

and the concept of heating food by microwaves was discovered when working on radar tubes.

Our approach

Inspiration for this thesis comes from an uncommon optimization domain. Picbreeder [85] is an online service that allows users to collaboratively evolve images via an artificial evolution. The authors observed that evolved images of particular complexity could not be directly re-evolved from scratch using the desired image as an objective function [104]. Our hypothesis is that users interacting with Picbreeder are inspired by initial images, define an objective function, and start to evolve towards it. However after few generations, the user might be inspired by the result to switch to another objective function. This process might be repeated several times, each time with a different objective function. We can illustrate this behaviour on one particular user reported by the authors of Picbreeder [92]. This user wanted to evolve an alien face and looked for similar images. After few generations, the user was surprised by one image which had transformed by random mutations into an image representing a car. The user found it more interesting and continued to rather evolve images similar to the car than the alien face. In this process, users have in advance no clear idea about what the results will be.

Our hypothesis is that the previously described behaviour of the users interacting with Picbreeder depicts a "creative process". For instance, artists or researchers might work by a similar process. The artist may have an initial vision or can start from scratch to see where it leads. Also, the researcher usually begins with an initial thought or an inspiration. It can be considered as an objective function. They develop their works towards their visions. After some time, they might be inspired by the result, which surpasses the original vision, and they change their goals and visions. In other words, they change their objective functions. Their results might represent something they would not have thought about before or the results might remotely differ from the previous or initial vision. This change might occur several times during the process.

In this thesis, we introduce a general framework on the top of a common optimization technique called inspiration-triggered search (ITS). It mimics the described behaviour, which we call a creative process, and produces "complex" solutions. We introduce one general instance of ITS and its concrete implementation for our testing domain. We test our approach in the domain of images, that is to find "complex" and aesthetically pleasant images for humans.

How to define a complex image is a difficult problem by itself. We use two different approaches for our evaluation. Firstly, we define that an image is complex when it cannot be found by the used optimization technique using the image as the objective function. However, we will show that even visually simple images are not able to be found by this approach. Secondly, we use machine learning methods to train a classifier to distinguish "simple" and "complex" images.

Many different approaches have been proposed to create interesting and aesthetically pleasant images for humans. The main problem is that the objective function is difficult to define formally because it is subjective. Examples of such approaches are evolutionary art [76], in which an artificial evolution is used to create works of art, and recent works [106, 97, 53, 107] based on visualizations of

deep neural networks or convolutional deep neural networks. We achieve interesting and aesthetically pleasant images by a choice of features that are used by the optimization technique.

At the first glance, our testing domain may seem to have only few similarities with more real-case usage of optimization techniques. Nevertheless, local optima and subjectivity are often shared by complex domains such as evolutionary robotics [65]. For instance, let us assume, the previously mentioned example of evolving a neural network that would allow a mobile robot to avoid obstacles. Firstly, it is hard to evolve the desired behaviour due to many local optima. Additionally, without an incremental approach, objective functions often introduce the bootstrap problem: if all individuals from the first randomly generated population perform equally poorly, the evolutionary process will not generate any interesting solution. Secondly, behaviours with the same score may be subjectively completely different. For instance, a robot exploring the environment will be labelled as more "intelligent" by humans in comparison with a robot moving in a circle, even though both of them may have the same score.

The thesis is dived into four chapters. The first chapter reviews foundational and related work for this thesis. In the second chapter, we propose an inspiration-triggered search. Firstly, we introduce it as a general framework. Then, we introduce its general instances by including several general ideas. Lastly, we present the concrete definition for our testing domain of images. The third chapter describes settings of our experiments. In the last chapter, we evaluate the proposed method.

1. Background and related work

This chapter reviews foundational and related work for this thesis. Firstly, we review artificial neural networks and compositional pattern producing networks that are used for our image representation. Secondly, we review techniques from the field of evolutionary computation that we use as our underlying optimization techniques. We briefly introduce interactive evolutionary computation and evolutionary art that are highly connected with our testing domain of images. Thirdly, we need to choose features for the underlying optimization technique, therefore we have a look what has been used in evolutionary art and we describe several measures from computational aesthetics and focus measures. Then, we briefly present Picbreeder which was the source of our initial inspiration. Lastly, we present other approaches dealing with the same topic as this thesis such as novelty search and computational creativity.

1.1 Artificial neural network

Artificial neural networks (ANNs) [2] are a family of computational models inspired by biological neural networks representing a system of interconnected nodes called neurons. For more information, please see the current literature.

1.1.1 Compositional pattern producing network

Compositional pattern producing networks (CPPNs) introduced by Stanley [90, 89] are a variation of artificial neural networks that differ in their set of activation functions. The choice of functions for the canonical set can be biased towards specific types of patterns and regularities.

According to Stanley, CPPN is an abstraction of natural development. Patterns in nature can be described at a high level as compositions of functions, wherein each function in the composition represents a stage in development. Thus the indirect CPPN encoding can compactly encode patterns with regularities such as symmetry, repetition, and repetition with variation.

CPPNs have shown promise in a variety of different domains including the evolution of two-dimensional images [85], three-dimensional objects [14], musical compositions [35], dancing avatars [21], robot morphologies [71, 12, 3], and connectivity patterns of complex neural networks [91].

1.2 Evolutionary computation

Evolutionary computation [22] is a collective name for a range of population-based metaheuristic optimization algorithms based on principles of biological evolution, such as natural selection and genetic inheritance.

1.2.1 Evolutionary algorithm

Evolutionary algorithms (EAs) [22] are based on adopting Darwinian principles and involve techniques implementing mechanisms inspired by biological evolution

such as reproduction, mutation, recombination, and natural selection. Although EAs can be divided into a number of branches, they all follow the same general framework. A cycle of evaluation, selection, and mutation is applied repeatedly to shape a population with respect to a fitness function. For instance, genetic algorithm approaches generally optimize strings of numbers that represent parameters in a problem domain [25]; genetic programming techniques evolve computer programs as trees of operators and operands to solve computational tasks [40]; and neuroevolution techniques evolve the structure and connection weights for artificial neural networks to perform control and decision-making tasks [28, 94].

1.2.2 NeuroEvolution of augmenting topologies

NeuroEvolution of Augmenting Topologies (NEAT) [94, 95] is a neuroevolution technique to evolve artificial neural network topologies along with weights. It is based on applying three key techniques: tracking genes with historical markings to allow crossover among topologies, protecting structural innovation using speciation, and incrementally growing from minimal structure. Although NEAT was originally introduced to evolve ANNs, it is sufficiently general to evolve CPPNs. The variation called CPPN-NEAT was introduced in [90].

1.2.3 Interactive evolutionary computation

Interactive evolutionary computation (IEC) [98] is well-suited for domains where a fitness function is subjective or difficult to define formally. For example, traditional evolutionary algorithms would struggle to determine whether an image is "attractive" or not, yet humans can easily perform such evaluations.

In single-user interactive evolution, the user is presented with a set of alternatives generated by the system. This initial population is then evolved over generations. In each generation, the user selects the most promising designs, that are then mated and mutated to create the next generation. In effect, IEC assists the user in exploring a potentially vast design space.

A prominent application of IEC is evolutionary art, in which the artificial evolution is used to create works of art. It has been successfully applied in a wide variety of application domains including the evolution of images [87, 88, 100, 77, 50], music [61, 9, 37], three-dimensional models [64, 36], dancing avatars [21], particle systems [32], and movies [101].

While IEC is a powerful approach for helping users to generate digital artefacts, results are often limited by human fatigue [98]. According to Takagi [98], a normal IEC process should only require 10 to 20 generations from the user. However, it is challenging to produce notable artefacts within this limit.

Many researchers have tried to get rid of human beings from IEC or at least partially simulate them. One approach is to use machine leaning techniques to learn the user preferences. For instance, a work by Johanson and Poli [37] learns a neural network when the user is interacting with the system. The trained neural network is afterwards used as a fitness function. Other approach is to formalize user preferences. For instance, computational aesthetics (see 1.3) try to describe the beauty in domains of human creative expression. A study by Lehman and Stanley [46] uses impressiveness defined by rarity and re-creation effort, that is

the difficulty for the benchmark optimizer to re-create an observed property of an evolved artefact. A recent study by Woolley and Stanley [105] uses novelty search (see 1.6) to get rid of a fitness function.

1.2.4 Evolutionary art

Evolutionary art is a research field where methods from evolutionary computation are used to create works of art. For a good overview of this field, see [76, 8].

The creation of fitness functions for the evaluation of art is regarded as one of the open problems in evolutionary art [57]. Within the field of evolutionary art, there are two main approaches to assign a fitness value to an artefact: to delegate fitness assignment to a human being [87, 77] or to use an automatic fitness assignment [5, 19]. To overcome the disadvantages and also combine the advantages of both approaches, Machado et al. propose partially interactive evolution [51], where the human user's contribution is much reduced compared to the fully interactive approach, but the human still guides the evolution. Substantial efforts in evolutionary art research have been dedicated to studying and devising good aesthetic measures. A study by Li and Hu [47] suggests using machine learning techniques to learn the differences between aesthetic and non-aesthetic images.

Johnson [38] presents a taxonomy of usage of a fitness function in evolutionary art and music based on fitness scope and fitness basis. The fitness scope, a classification of what the fitness is applied to, is divided into three classes: set of works where each member of the population consists of a collection of individual artworks and the fitness measure is applied to that collection; each member represents the single artwork and the fitness is applied to a single work; and evolutionary process as artwork is where the fitness evaluation is part of a process which is viewed in some fashion as the work itself and therefore the work as such is not being rated by the fitness measure. The fitness basis, a classification of how the fitness is evaluated, is divided into five classes: aesthetic measure (a fixed function measuring the quality of the solution); human interaction; the use of a corpus of material or guiding example; an endogenous or implicit fitness derived from interactions between agents; and the use of a population of critics that learn alongside the evolutionary process.

1.2.5 Image representation

In the field of evolutionary computation, there are many possible representations of genotypes, such as strings of binary digits, sets of procedural parameters, or symbolic expressions. The widely-used representations for images are tree structures and artificial neural networks. It is worth noting that the resulting images from these representations have possible infinite resolution.

Artificial neural network

The image is described by a function encoded in a form of an artificial neural network. The basic network's topology to create a black-and-white two-dimensional image contains at least two input nodes and one output node. Network's inputs represents X and Y coordinates of the pixel and the network's output value is a value for the pixel at position (X, Y). In order to create the image, we iterate

through each pixel at the image. We set coordinates of the pixel as inputs to the network, evaluate the network, and use the network's output as the value for this pixel. To create coloured images, the network usually contains three output nodes to create the pixel value in a colour space, for instance, such as RGB (red, green, blue) or HSB (hue, saturation, brightness). Restrictions of a network's topology and its activation functions create biased choice towards resulting images.

For instance, Picbreeder (see 1.5) creates coloured images represented by CPPNs. The network has three input nodes: X and Y coordinates of the pixel, and the distance from the centre of the image. A colour is represented in the HSB colour space, therefore the network has three output nodes. CPPNs include cosine, sine, Gaussian, identity, and sigmoid functions to represent the images. The network's topology is unconstrained and can represent any possible relationships (including recurrent).

Tree structure

The image is described by a formula or a computer program encoded in a form of a tree structure. This representation often relies on genetic programming. As in the previous representation, the choice of a function biases the resulting images. These functions can vary from basic mathematical functions, such as addition, division, modulo, sine, and cosine, to more complex functions, such as to blur an image or to create fractals. Examples of such approaches are [87, 77, 50].

1.3 Computational aesthetics

"Computational aesthetics is the research of computational methods that can make applicable aesthetic decisions in a similar fashion as humans can" [62]. In other words, it tries to describe the beauty in domains of human creative expression such as music, visual art, poetry, and others. For good overview of this field, see [30, 62].

Birkhoff [10] was the first to formalize quantitative theory of aesthetics. He proposed the aesthetic measure M defined as

$$M = \frac{Order}{Complexity} \tag{1.1}$$

It represents the perceptual reward for the effort of focusing attention on something complex but then realizing a certain pleasant harmony. From Birkhoff's work, a number of researchers [58, 7] developed information aesthetics based on information theory. The concepts of order and complexity were formalized from Shannon's notion of information [16].

Scha and Bod [81] stated that in spite of the simplicity of these beauty measures, "if we integrate them with other ideas from perceptual psychology and computational linguistics, they may in fact constitute a starting point for the development of more adequate formal models."

1.3.1 Machado & Cardoso

This measure is based on the aesthetic theory of Machado and Cardoso [49]. The aesthetic value of an artwork is related to the relation between image complexity

(IC) and processing complexity (PC). Images that are visually complex, but are easily processed have the highest aesthetic value. For an example, the authors refer to fractal images which are visually complex but can be described by a simple formula. The aesthetic measure M_{MC} of an image I is defined as

$$M_{MC}(I) = \frac{IC(I)}{PC(I)} \tag{1.2}$$

The image complexity can be estimated as the effort needed to compress the image and is defined as

$$IC(I) = \frac{RMS(I)}{CompressionRatio(I)} \tag{1.3}$$

where RMS is the difference between the original and the compressed image expressed as the root mean square. The compression ratio is the ratio between the original and the compressed image size. The authors suggest to use JPEG compression. The processing complexity can be calculated using fractal image compression [84].

1.3.2 Information entropy

This measure based on the Shannon entropy [16] is similar to the measure used in [70]. The measure M_{IE} of an image I is defined as

$$M_{IE}(I) = -\sum_{x \in X} p(x) \log(p(x)) \tag{1.4}$$

where X represents bins of the image brightness histogram with probability distribution p. For instance, a work [47] uses the histogram with 100 bins.

1.3.3 Benford law

This aesthetic measure is based on Benford law [39, 1]. Benford law (first-digit law) states that list of numbers obtained from real life source of data, not created by man, are distributed in a non-uniform way. The leading digit occurs one third of the time, the second digit occurs 17.6%, etc.

Benford law is used to measure the distribution of brightness of pixels. For instance, a study [34] calculates the brightness histogram using 9 bins. The measure M_{BL} of an image I calculated as

$$M_{BL}(I) = \frac{D_{max} - D_{total}}{D_{max}} \tag{1.5}$$

represents the difference between the actual histogram and the Benford histogram. D_{total} is computed as

$$D_{total} = \sum_{i=1}^{9} \left(\frac{H_{image}(i)}{N} - H_{benford}(i) \right)^p \tag{1.6}$$

where $H_{image}(i)$ is the number of entries in the brightness histogram bin number i and N is the total number of pixels in the image. $H_{benford}(i)$ is the value from

the Benford distribution and it is distributed for 9 bins as follows: 30.1%, 17.6%, 12.5%, 9.7%, 7.9%, 6.7%, 5.8%, 5.1%, and 4.6%. The maximal difference D_{max} is calculated as $(1 - 0.301)^p + (0.176)^p + \ldots + (0.046)^p$. This study [34] suggests to use $p = 1$.

1.3.4 Global contrast factor

The aesthetic measure proposed in [56] for grayscale images computes contrasts, the average difference between neighbouring pixels, at various resolutions. To build a human perception based method, the authors rather use a perceptual luminance than a pixel value. The perceptual luminance L for the original pixel value $k \in \{0, \ldots, 255\}$ is approximated as

$$L = 100 \cdot \sqrt{\left(\frac{k}{255}\right)^{\gamma}} \tag{1.7}$$

with gamma correction $\gamma = 2.2$.

The local contrast c_i for pixel i is the average difference in perceptual luminance between the pixel and four neighbouring pixels. Assuming the image is w pixels wide, h pixels high, and the image is organized as one-dimensional array of row-wise sorted pixels, the local contract c_i for pixel i is computed as

$$c_i = \frac{|L_i - L_{i-1}| + |L_i - L_{i+1}| + |L_i - L_{i-w}| + |L_i - L_{i+w}|}{4} \tag{1.8}$$

For pixels at the edges, only the available neighbouring pixels are taken into account. The average local contrast C for the current resolution is computed as the average local contrast over the whole image.

$$C = \frac{1}{w \cdot h} \sum_{i=1}^{w*h} c_i \tag{1.9}$$

The image is made smaller by combining a fixed number of pixels into one super pixel. The super-pixel value is computed as the average of original pixel values. They use super-pixels of following sizes: 1 (the original image), 2, 4, 8, 16, 25, 50, 100, and 200 resulting in 9 different resolutions.

The measure M_{GCF} for an image I is defined as

$$M_{GCF}(I) = \sum_{i=1}^{9} (w_i \cdot C_i) \tag{1.10}$$

where C_i is the average local contrast for the resolution given by the size of super-pixel at index i from the previously mentioned set. They experimentally defined weighting factors w_i as

$$w_i = (-0.406385 \cdot \frac{i}{9} + 0.334573) \cdot \frac{i}{9} + 0.0877526 \tag{1.11}$$

where $i \in \{1, \ldots, 9\}$.

1.3.5 Relaxed symmetry

The aesthetic measure proposed in [33] computes a reflectional symmetry of an image. The image is divided in four quarters cutting the image in half across the horizontal and vertical axis. Let us denote A_1 as top left quarter of the image, A_2 as top right, A_3 as bottom left, and A_4 as bottom right quarter of the image.

The horizontal symmetry S_h of the image I represents the average similarity between left and right half of the image and is defined as

$$S_h(I) = s(A_{left}, A_{right}) \tag{1.12}$$

where $A_{left} = A_1 + A_3$, and $A_{right} = A_2 + A_4$. The vertical symmetry is calculated as

$$S_v(I) = s(A_{top}, A_{bottom}) \tag{1.13}$$

where $A_{top} = A_1 + A_2$, and $A_{bottom} = A_3 + A_4$. The diagonal symmetry is defined as

$$S_d(I) = \frac{s(A_1, A_4) + s(A_2, A_3)}{2} \tag{1.14}$$

The similarity between two areas A_i and A_j is defined as

$$s(A_i, A_j) = \frac{1}{w \cdot h} \sum_{x=0}^{w} \sum_{y=0}^{h} sim(A_i(x, y), A_j^m(x, y)) \tag{1.15}$$

where x and y are the coordinates of the pixel, w is the width, and h is the height of the area. A_j^m is the mirrored area of A_j. For horizontal symmetry A_j is mirrored around the vertical axis, for vertical symmetry A_j is mirrored around the horizontal axis, and for diagonal symmetry A_j is mirrored around both axes. The similarity between two opposing pixels is defined as

$$sim(A_i(x, y), A_j(x, y)) = \begin{cases} 1 & \text{if } |I(A_i(x, y)) - I(A_j(x, y))| < 0.05 \\ 0 & \text{otherwise} \end{cases} \tag{1.16}$$

where $I(A_i(x, y)) \in [0, 1]$ is the intensity value of the pixel (x, y) in area A_i.

The relaxed symmetry measure M_{RS} for the image I is defined as

$$M_{RS}(I) = e^{-\left(\frac{(M_{SS}(I) - 0.8)^2}{0.08}\right)} \tag{1.17}$$

where M_{SS} is the combined symmetry calculated as

$$M_{SS}(I) = \frac{S_h(I) + S_v(I) + S_h(I)}{3} \tag{1.18}$$

1.4 Focus measures

Another important measure of an image is a sharpness. In literature [42, 48], a focus measure has a maximum value for the best focused image and it generally decreases as the defocus increases. For overview of focus measures, please see [68].

1.4.1 Tenengrad

Tenengrad focus measure M_T [79, 96] for an image I, based on the magnitude of an image gradient, is defined as

$$M_T(I) = \sum_{(i,j)\in I} \left(G_x(i,j)^2 + G_y(i,j)^2\right) \tag{1.19}$$

where G_x and G_y are the X and Y image gradients computed by convolving the image I with the Sobel operators. They use a kernel size of 3×3, that means

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I \tag{1.20}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I \tag{1.21}$$

where $*$ denotes the 2-dimensional convolution operation.

1.4.2 Normalized variance

The normalized variance [79, 96] is defined as the variance divided by the mean of an image. The measure M_{NV} for an image I is calculated as

$$M_{NV}(I) = \frac{\sigma^2}{\mu} \tag{1.22}$$

where σ is the standard deviation and μ is the mean of image pixel values.

1.5 Picbreeder

Picbreeder [85] is an online service that allows users to collaboratively evolve images via an artificial evolution. Users can begin evolving from scratch as in typical interactive evolutionary computation applications or from already published images by branching, thereby continuing its evolution. The authors observed that evolved images of particular complexity could not be directly re-evolved from scratch using the desired image as an objective function [104].

Coloured images are represented by CPPNs (see 1.1.1). The network has three input nodes: X and Y coordinates of the pixel, and the distance from the centre of the image. A colour is represented in the HSB colour space, therefore the network has three output nodes. CPPNs include cosine, sine, Gaussian, identity, and sigmoid functions to represent the images. These functions were chosen to capture regularities that appear frequently in nature (e.g. symmetry, repetition, repetition with variation) without an intentional aesthetic bias. The topology of network is unconstrained and can represent any possible relationships (including recurrent). The images are evolved by NEAT (see 1.2.2).

1.6 Novelty search

When a fitness function is highly deceptive [26], it may misdirect the search process towards dead-ends instead of guiding it. Lehman and Stanley [43, 45] argue that, in these cases at least, it may be useful to get rid of the fitness function and only search for novel behaviours. They also argue that this radical departure from objective-based search may be a better abstraction of how natural evolution continuously "discovers" new lifeforms. They point out that sometimes not looking for the goal in this way leads to finding the goal more quickly and consistently. Lehman and Stanley exploited this idea in novelty search [43, 45]. It is an evolutionary process in which solutions are compared according to their behaviours and ranked according to their novelty with regard to all the behaviours that have been discovered before.

Novelty search replaces the fitness function with a novelty metric, which is a user-defined measure of distance between evolved artefacts in a particular domain. They propose the novelty metric based on the average distance to the k-nearest neighbours of what the search has previously encountered.

The authors showed that novelty search is an effective algorithm for deceptive two-dimensional robot maze navigation task and for evolving a neuro-controllers for bipedal walking. Other authors have confirmed that novelty search is a promising alternative to objective-based search [72, 41]. Novelty search was also used in interactive evolutionary computation [105] and open-ended evolution [46]. Open-ended evolution is defined as "a process in which there is the possibility for an indefinite increase in complexity" [78]. Lehman and Stanley proposed a modification called minimal criteria novelty search [44], in which individuals must meet domain-dependent criteria to be selected for reproduction. Novelty search can be combined with the objective-based search thanks to multi-objective EAs: one objective is the traditional fitness function, the other is the novelty score used in novelty search [60].

Novelty search raises a new question: in what space do we measure novelty? For instance, a recent work [63] replaces the human-crafted behavioural distance with a deep neural network that can recognize interesting differences between phenotypes. Another approach is to formalize interestingness and define a curiosity-driven search process [66, 83]. Curiosity is the drive to actively explore the interesting regions in the search space that most improve the model's predictions or explanations of what is going on in the world.

1.7 Computational creativity

Computational creativity [15, 54] is a multidisciplinary research located at the intersection of the fields of artificial intelligence, cognitive psychology, philosophy, and the arts. It is the study of building software that exhibits behaviour that would be deemed creative in humans. Such creative software can be used for autonomous creative tasks, such as inventing mathematical theories, writing poems, painting pictures, and composing music. However, computational creativity studies also enable better understanding of human creativity and to produce programs for creative people to use, where the software acts as a creative collaborator rather than a mere tool.

2. Inspiration-triggered search

In this chapter, we propose an inspiration-triggered search (ITS). Firstly, we introduce ITS as a general framework. Then, we introduce its general instances by including several general ideas. Lastly, we present a concrete definition for our testing domain of images. Even though, ITS is a general framework, we use terminology from the field of evolutionary computation (see 1.2) because our main focus is on this field.

2.1 Framework of inspiration-triggered search

In this section, we introduce a general framework of inspiration-triggered search (ITS). Similarly to the field of evolutionary computation, evolutionary algorithm (see 1.2.1) is a general framework as well. Examples of its instances are evolutionary algorithms, genetic programming, and neuroevolution.

2.1.1 Inspiration

Picbreeder (see 1.5) is an online service to collaboratively create images via an artificial evolution. The authors observed that users evolved images with particular complexity that could not be directly re-evolved from scratch using the desired image as an objective function. Our hypothesis is that users are inspired by initial images, define an objective function, and start to evolve towards it. However after few generations, they might be inspired by the results to switch to another objective function. This process might be repeated several time, each time with a different objective function. We can illustrate this behaviour on one particular user reported by the authors of Picbreeder [92]. This user wanted to evolve an alien face and looked for images similar to it. After few generations, the user was surprised by one image which had transformed by random mutations into an image similar to a car. The user found it more interesting and continued to rather evolve images similar to the car than the alien face. In this process, users have in advance no clear idea about what the results will be.

Our hypothesis is that the previously described behaviour depicts a "creative process." For instance, artists and researchers might work by a similar process. The artist can have an initial vision or can start from scratch to see where it leads. Also, the researcher usually begins with an initial thought or an inspiration. It can be considered as an objective function. They develop their works towards their visions. After some time, they might be inspired by the result, which surpasses the original vision, and they change their goals and visions. In other words, they change their objective functions. The results might represent something they would not have thought about before or the results might remotely differ from the previous or the initial vision. This change might occur several times during the process.

2.1.2 General idea

Inspiration-triggered search (ITS) mimics the described behaviour on the top of a standard optimization technique. One can choose any optimization technique. It can be population-based or without a population, that is, it operates only with one individual. Nevertheless, we describe ITS as if it uses a population-based optimization technique, but the population can of course contain only one individual.

ITS generalizes the described behaviour into three phases: the inspiration phase, the optimization phase, and the diversification phase. In the inspiration phase, the current population is analysed to detect "interesting or promising features" and a new objective function is defined. In other words, the most "interesting or promising" individuals are used to define the new objective function. For instance, the reported user from Picbreeder wanted to evolve an alien face. However, the user visually analysed the population, found a more interesting image representing a car, and defined the new objective function to evolve the car. The optimization phase represents the standard optimization using the current objective function. The underlying optimization technique will end if either it converges or after a pre-defined number of iterations. If ITS is "inspired" by an individual which has been found, it will define the new objective function by switching to the inspiration phase. Otherwise, if the optimization technique has converged, ITS will switch to the diversification phase. The idea of the diversification phase is to add diversity to the population. It is mainly used after the convergence of the optimization technique. ITS diversifies the population until something "inspiring" in the population is found. Then, it defines the new objective function and switches back to the optimization phase.

2.1.3 Algorithm

Inspiration-triggered search starts with a random population and defines an initial objective function from this population. This can be considered as a special case of the diversification phase, therefore ITS does not need any initialization phase. Algorithm 1 shows a pseudo-code of ITS.

During the diversification phase, the *diversify* method diversifies the population until something "inspiring" in the population is found, which is detected by the inspiration criterion, and then ITS switches to the optimization phase. The *diversify* method can change a size of the population. For instance, it might be useful for the inspiration criterion to create more individuals. The size can later be changed by the *select* method, which prepares the population for the underlying optimization technique. If the population is diversified and the inspiration criterion is not met, it could mean that the objective function is too "complex" and the population does not contain any individual close to this objective function. The objective function can be simplified by the *simplify* method.

In the inspiration phase, the *inspire* method defines the new objective function from the current population. The following *select* method prepares the population for the optimization technique. For instance, it can choose a subset of the population or modify it.

In the optimization phase, ITS optimizes the population by the underlying optimization technique using the current objective function. We have only one

Algorithm 1 The general framework of inspiration-triggered search

```
1: procedure INSPIRATION-TRIGGERED SEARCH
2:     P ← ∅                                              ▷ Population
3:     f ← ∅                                              ▷ Objective function
4:
5: diversification:                                       ▷ Diversification phase
6:     P ← diversify(P)
7:     if inspiration criterion is met then
8:         goto inspiration
9:     else
10:        f ← simplify(f, P)
11:    goto diversification
12:
13: inspiration:                                          ▷ Inspiration phase
14:    f ← inspire(f, P)
15:    P ← select(P)
16:    goto optimization
17:
18: optimization:                                         ▷ Optimization phase
19:    optimization technique with f and P for at most k iterations
20:    if inspiration criterion is met then
21:        goto inspiration
22:    else if convergence criterion is met then
23:        goto diversification
24:    goto optimization
```

requirement for the optimization technique. It must be able to run in a specific number of iterations denoted by $k \in \mathbb{N}$, which depends on the convergence and the inspiration criterion. Otherwise, one can choose any optimization technique. The optimization technique will end if either it converges for a pre-defined number of generations or after k iterations. In other words, the population is checked for "interesting or promising" individuals every k-th iteration. The inspiration criterion is used to detect whether any individual is enough "interesting or promising" to change the objective function. If the inspiration criterion is met, ITS will switch to the inspiration phase. Otherwise, the convergence criterion is used to detect a premature convergence of the optimization technique. If it is met, ITS will switch to the diversification phase.

2.1.4 Properties

The definition of the objective function, the inspiration criterion, and the *inspire* method are main building blocks needed to be defined for ITS. Nevertheless, we can deduce a few general properties of ITS. Firstly, the current population is crucial to the search. It would not work if the population was forgotten after a definition of the objective function and ITS started to optimize from scratch. The objective function should be defined by the current population. Therefore, the population contains some individuals that do not perform poorly according to the

objective function. This way, the bootstrap problem is not introduced into the search: if all individuals from the first randomly generated population perform equally poorly, the evolutionary process will not generate any interesting solution. Without the current population and with a "complex" objective function, the search would have to deal with the bootstrap problem and it could prematurely converge. A different point of view is that ITS is an automated incremental approach. It can start with a simple objective function and continuously make it more complex.

ITS operates in cycles that we call creative cycles. A creative cycle contains the optimization using the current objective function and it can be followed by the inspiration or the diversification phase resulting in the definition of the new objective function. The initial diversification phase and the definition of the initial objective function can be considered as the initial phase. After this initial phase, ITS runs in creative cycles. One creative cycle can be considered as one run of the optimization phase, that are lines 18 to 24 in the algorithm 1.

Before introducing concrete definitions, we explain the main idea of ITS for our domain of images. Proposed instances of ITS will follow an incremental approach. We try to gradually improve what has been found. We do not want to make a rapid changes to the objective function. We rather let the search gradually adapt to the changes. Let us assume, the image contains one circle and the objective function is somehow defined as a circle in that area. After few generations, a second circle has appeared in the image during the optimization. We also add the second circle to the objective function. Again, after few generations, two lines have appeared in the image and both circles have disappeared. We add one of lines into the objective function and remove one circle. This way, we look for one circle and one line in images. This example illustrates graduate changes of the objective function. Furthermore, we try to preserve some information from the previous objective function, in this example, one circle. We do not want to lose this information. Nevertheless, one can change the desired behaviour for different needs in particular domains. For instance, one could change the objective function to reflect more "promising" images, that is not preserving information from previous objective functions. We rather modify the objective function step by step. From now on, we will keep this idea in mind for following definitions.

2.2 Description-based ITS

In this section, we introduce a general instance of the previously presented ITS framework. Even though this instance is still general, we mainly explain it by examples from our testing domain of images.

2.2.1 General idea

We can follow the previously described example of the reported user from Picbreeder who wanted to evolve an alien face. Obviously, we would like to somehow define the objective function as the alien face. Let us assume, we can get information about "good features" from an image. In our example, we would like get information about objects in the image, for instance, a list of objects with their prediction accuracies. Even for humans, the first appearance of a car could have

been far away from a typical representation of the car. We call this information a description of a particular image. We are able to get this information and define the description for each individual from the population. Therefore, the population of individuals is interchangeable with the population of descriptions. In other words, we can almost forget about the individuals and focus only on the descriptions.

We introduce a user-defined description metric in a particular domain between the objective function and the description of the particular individual. This metric can be useful to define the inspiration criterion and the *inspire* method. We propose that the inspiration criterion will be met if the description metric value is bigger than a pre-defined threshold. In our previous terms, we called this image "interesting, promising, or inspiring."

As a side remark, one could define the description metric similarly to novelty search (see 1.6), that is, the description metric would be defined between two descriptions. This way, the search could search for novel descriptions.

2.2.2 Definition

Let us assume, we are able to get a description $D \in \mathbb{D}$ for an individual $I \in \mathbb{I}$ by a function $\bar{D} : \mathbb{I} \to \mathbb{D}$. Let us denote a population P of individuals, an objective function $f \in \mathbb{O}$, and a description metric $\delta \colon \mathbb{O} \times \mathbb{D} \to \mathbb{R}$ between the objective function and the description.

We propose the inspiration criterion by the condition

$$\exists I \in P : \delta\left(f, \bar{D}(I)\right) \geq t \tag{2.1}$$

for a threshold $t \in \mathbb{R}$.

2.3 Multi-view ITS

In the previous section, we mentioned the example of the reported user from Picbreeder who wanted to evolve the alien face. Obviously, we would like to somehow define the objective function as the alien face. A high-level approach could be to use a "database" or a pre-trained classifier for an object recognition. For instance, one could use the current state of the art methods based on deep learning [18]. However, we use a low-level approach with pre-defined feature functions for a particular domain which can be considered as a classical optimization approach.

2.3.1 General idea

In the field of interactive evolutionary computation (see 1.2.3), human beings are used to rate individuals. We look at the problem from a different perspective. Instead of asking human beings to rate individuals, we want them to label "interesting or promising" parts of individuals. For instance, in the domain of images, it could be visually pleasant areas or areas partially resembling faces. Instead of using interactive evolutionary computation, we can relax humans algorithms. For instance, in our domain, we can label objects in images. This should represents a

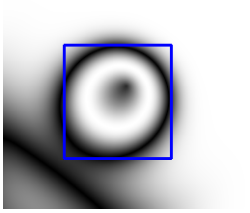


Figure 2.1: Example of the description defined by a detection of objects in the image.

similar behaviour as in visual areas of human brains [6]. For instance, the figure 2.1 shows a possible "promising" area of the image.

From a different point of view, we obtain these parts from a supervisor. This "hypothetical" supervisor can be an expert in one field and we receive the expert's opinion. For instance, teachers of art schools may go to each student in a classroom and tell the student which part of the painting looks good and which part the student should focus on. The parts may still looks bad and immature. But with continuous work, the final result may be way better and maybe even a masterpiece might arise. Thus, we call these parts promising.

Nevertheless, we only somehow obtained information about the parts and we do not know why they are supposed to be promising. For instance, students might be in a similar situation. They receive feedbacks from their teachers about their works, for instance, such as paintings, compositions, and essays. The teachers are more experienced and they can see further away than their students. This way, the students do not receive particular information why the parts are promising. Maybe, the teachers are not even able to explain their reasons. Therefore, the students can only improve their works by the only things they know and they understand. We use the same analogy with pre-defined features for a particular domain. We try to improve the population by the only things we can do: the optimization of pre-defined features.

For simplicity, let us assume, we have only one pre-defined feature and we are given one promising part of the image. We can only do one thing in this context. We focus on optimizing the feature in this part of the image, for instance, as in the figure 2.1. With more promising parts, we can use the same principle. We optimize the feature in a subset of areas. We call these parts windows. Thus, we are given promising windows in the image. In this sense, the objective function is defined as a collection of windows to optimize.

By including more pre-defined features, we must determine a feature to optimize for a particular window. We found more convenient and robust to rather think about it as "windows in features" than "features in windows." This way, each feature has a collection of windows to optimize. To avoid any problems, let us assume that features values do not depend on the size of the window. The objective function consists of a collection of features, that we call views, and each view consists of a collection of windows. The search optimizes these windows. Each view and each window is associated with a particular feature function. The figure 2.2 shows a visual example of this representation of the objective function.

Until now, we assumed the existence of only one supervisor who gives us promising parts. Nevertheless, we may have more experts and each of them

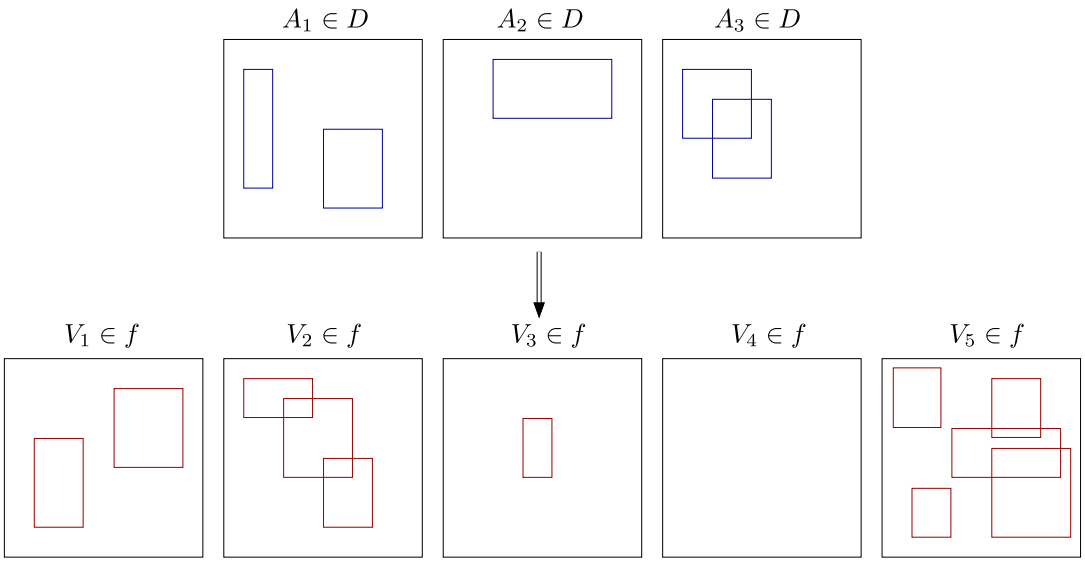


Figure 2.2: Example from the domain of images of visual representation of projecting windows from the description $D = \bar{D}(I)$ of the particular individual I to the objective function f. Blue rectangles are windows representing promising areas in the image. Views A_1, A_2, and A_3 represent different viewpoints for the image. Red rectangles are windows that are currently optimized. Views V_1, V_2, V_3, V_4, and V_5 represent the pre-defined features.

can tell us promising parts from a different point of view. Again, we call these parts windows and the expert's opinions views. The collection of views is the description of the particular individual presented in the previous section. The figure 2.2 shows a visual example of this representation of the description. To summarize, we are able to obtain the description for the particular individual. This description consists of a collection of views representing different experts viewpoints. Each view consists of a collection of windows representing promising parts from this viewpoint.

The objective function can be considered as a special kind of a description of an individual we search for. It has the same structure. Nevertheless, the description of the individual and the objective function may have different dimensions. Precisely, they consist of a different number of views. It basically means we try to project windows of views from the description into views from the objective function, see figure 2.2. We must not forget that they represent something else, experts opinions versus features spaces.

2.3.2 Properties

We call the presented approach multi-view ITS. We obtain promising parts from the different point of views and we try to project them into features spaces. In the field of semi-supervised learning [2], which makes use of labelled and unlabelled data, there is an area called multi-view learning. The idea of multi-view learning is to look at an object from more different viewpoints. Nevertheless, the general idea is the only shared property with our approach because we do not deal with semi-supervised learning.

From another point of view, a window in the objective function defines a new

"local" feature. This local feature is defined by the pre-defined feature function but it is applied only to a sub-problem, for instance in our domain, an area of the image. From this viewpoint, the objective function is defined by local features and we create new local features based on building blocks defined by the pre-defined "global" features. In other words, the full problem is relaxed and we search the search space by solving sub-problems.

We believe that this approach can be applied to any difficult domain that can be split into sub-problems. Obvious application is art. For instance, in the domain of music, one can look for the parts with harmonies, beats, and so on. One can easily imagine other possibilities from art such as movies, books, and so on. Nevertheless, we believe it can even be applied to more traditional optimization problems. For instance, NP-hard problems such as maximum satisfiability problem (MAX-SAT): the problem of determining the maximum number of clauses, of a given Boolean formula in conjunctive normal form, that can be made true by an assignment of truth values to the variables of the formula. The description's views can represent special structures of clauses. Another example could be any graph problem, in which we are able to select sub-graphs with special properties.

In the field of evolutionary art (see 1.2.4), several works [74, 75, 52, 80] used critics or co-evolution of critics. Critics make comments and evaluate artworks, which can be used to change the artworks. The latter evolves a population of critics alongside the regular population. Nevertheless, they typically assign a single aesthetic value to an artwork.

2.3.3 Definition

Let us assume, we have N feature functions $F_1, \ldots, F_N$ for the particular domain and their values do not depend on the size of the problem in this domain. The window $W \in \mathbb{W}$ labels a part of the original problem. The view V is a collection of windows. The description D is a collection of views. The objective function f has the same structure as the description. For clarification purposes, we denote the window and the view as W_I, V_I when it comes from the description $D = \bar{D}(I)$ of the particular individual I and as W_O, V_O when it is part of the objective function f. F_{V_O} is the feature function associated with the view V_O from the objective function f. $F_{V_O}(W, I)$ is the value of the feature function F_{V_O} applied to the part specified by the window W in the individual I. A metric $\|D\|_W$ represents a number of windows in the description D and a metric $\|D\|_V$ represents a number of views in the description D.

2.3.4 Objective function for optimization technique

We have presented the structure of the objective function that is used by the underlying optimization technique. To recall, the objective function consists of windows for each feature (view) that the optimization technique optimizes. In other words, we have a feature value for each window. This represents a multi-objective optimization problem. Nevertheless, the most optimization techniques only use one value for an individual. To solve this problem, one could use scalarization approaches, in which multi-objective optimization problems are transformed to single-objective optimization problems, or approaches based on Pareto fronts. For

instance, in the field of evolutionary computation, one could use NSGA-II [17].

We use a basic linear scalarization. The fitness value for the individual I is computed as

$$\sum_{V_O \in f} \sum_{W_O \in V_O} (w_{W_O} \cdot F_{V_O}(W_O, I)) \tag{2.2}$$

where f is the current objective function, and $w_{W_O} \in \mathbb{R}$ is the weight for the window W_O. One could define the weight as the size of the window, that would result in preferring larger areas with high features values. In our domain, we do not prefer bigger areas, therefore each window W_O has the same weight of $w_{W_O} = 1$.

Another problem can be caused by using features with different domains. High values of one feature can decrease the meaning of other features. Furthermore, in our domain, we may not know the domain of features in advance. Even, the knowledge of the domain does not help when the values are irregularly distributed. For instance, a mathematical upper bound may be too high and no individual probably ever reaches this value. It would result in decreasing the meaning of this feature. Therefore we use adaptive scaling of features values. The idea is to scale each feature to the range of $[0, 1]$ by the previously observed values. Then, each feature has the same weight for the underlying optimization technique. The feature value is scaled by the maximum and the minimum value of this feature that has been found until the previous generation. The scaled value can be larger than 1 only when the new bound has been found this generation. The scaled value of the feature F with value x in the generation number $t + 1$ is defined as

$$\frac{x - F_t^{MIN}}{F_t^{MAX} - F_t^{MIN}} \tag{2.3}$$

where F_t^{MIN}, F_t^{MAX} is the minimum, the maximum value of the feature F until the generation number t.

2.3.5 Description metric by coverage and extension

In this thesis, we propose a description metric between the objective function f and the description D based on two parts: the coverage and the extension. For purposes of a demonstrative explanation, we assume that the description and the objective function contain only one view. The first part represents how much the objective function f is similar to or preserved by the description D. If the new objective function is defined as the description D, a high value of the coverage part should keep the similar "structure" of the previous objective function f. The latter part represent how much the description D extends the objective function f. If the new objective function is defined by adding the description D into f, a high value of the extension part should represent a higher "informational" gain in comparison with f.

One can think about it from a typical perspective of local search: exploitation and exploration. If the search focuses only on the coverage part, it will focus on exploitation. It forces to preserve what has found and it focuses on this area in the search space. On the other hand, if the search focuses only on the extension part, it will focus on exploration. It will search for different things in comparison with it has now. As in algorithms of local search, we need to find a right balance

between exploration and exploitation. With this formalization, one can explicitly set the focus on the desired part and set the desired balance between them.

Formally, the description metric δ is defined by the coverage $C : \mathbb{O} \times \mathbb{D} \to \mathbb{R}$ and the extension $E : \mathbb{O} \times \mathbb{D} \to \mathbb{R}$ between the objective function f and the description D as

$$\delta(f, D) = C(f, D)^{\alpha} \cdot E(f, D)^{\beta} \tag{2.4}$$

where $\alpha, \beta \in \mathbb{R}$ are parameters. We define it as a multiplication, because we want to focus on both parts. To recall, we would like to keep the presented gradual incremental approach. We prefer to have a good trade-off between the coverage and the extension than extreme cases. As a side remark, one could define that the inspiration criterion will be met if the extension value of any individuals is greater than a pre-defined threshold. In other words, the inspiration criterion would focus on the exploration of search space.

For later purposes, we found also convenient to define the extension and the coverage from the point of view of a single window. We denote $C(W_O, D)$ or $C(W_O, V_I)$ the coverage between the window $W_O \in f$ and the description D or the view $V_I \in D$, and $E(f, W_I)$ or $E(V_O, W_I)$ the extension between the objective function f or the view $V_O \in f$ and the window $W_I \in D$. Formally, they are the same functions because we can extend the window or the view into the full representation of the description.

2.3.6 Inspire method

The *inspire* method can be split into two parts. Firstly, we need to select the description(s) to define the new objective function. To make it simple, we can assume the selection of only one description. If we select more descriptions, we could make their intersection or union depending on a particular domain. The simplistic way is to select the description according to the description metric value. There are two basic options: to select the description with the highest value or to randomly select the description with the probability defined by its description metric value. Another option would be to only consider the individuals satisfying the inspiration criterion.

Secondly, after the choice of the description, we modify the current objective function by this description. There are three basic options to modify the objective function: adding a new window to the objective function, and modifying or removing an existing window from the objective function. For these modifications, we use general operators. For instance, evolutionary algorithms use general operators such as mutations and crossovers to modify the population. To make it simple, we propose operators that can change at most one window. This approach follows our gradual incremental approach. We do not make huge changes in the objective function that could be hard for the population to adapt to. For their definition, we use the previously defined extension and coverage between each window and the current objective function. In this thesis, we do not use operator to change the existing window.

Adding a window

We need to choose a window from the description D to add it to the objective function f. For this purpose, we use information about the extension for each window. We prefer the window with the higher extension value. To recall, it represents how much the window extends the objective function. Furthermore, we have extension values $E(V_O, W_I)$ between each window W_I from the description D and each view V_O from the objective function f. Therefore, we can immediately chose a view where to add a window. We make two decisions at one step: the choice of the window and the choice of the view representing the particular feature where to add this window. There are two basic options: to select the window with the highest extension value of to randomly select the window with the probability defined by its extension value.

Removing a window

We need to choose and remove a window from the objective function f. For this purpose, we use the information about the coverage of each window. To recall, it represents how much the window from f is preserved by the particular description. This can represents a support to keep the window in the objective function. In other words, $1 - C(W_O, D)$ is the probability to remove the window $W_O \in f$ from the objective function f.

Additionally, we introduce a threshold for a window to be removed and a limitation of removing only one window. That means, we iterate through all windows $W_O \in f$ in the objective function f. If the coverage value $C(W_O, D)$ is smaller than the threshold $R \in \mathbb{R}$, where D is the description used to modify the current objective function, the window will be removed with probability $1 - C(W_O, D)$. If the window is removed, we will end. We want to remove at most one window.

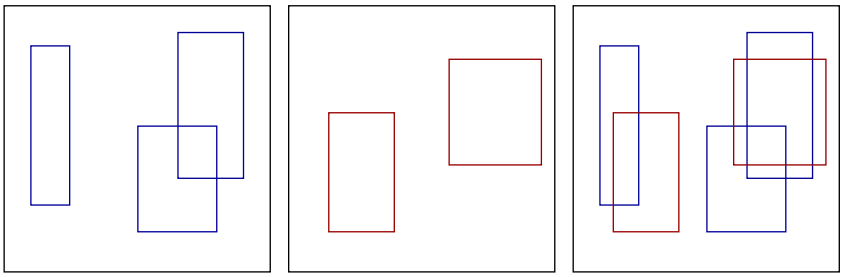
2.3.7 Simplify method

To recall, the *simplify* method can simplify the current objective function during the diversification phase when the inspiration criterion is not met. The simplistic approach is to remove one window from the objective function f each time the inspiration criterion is not met. We propose a simple method based on the average coverage of the window in the whole population. It computes the averaged coverage for the window $W_O \in f$ from the objective function f as

$$\frac{1}{|P|} \cdot \sum_{I \in P} C(W_O, \bar{D}(I)) \tag{2.5}$$

where P is the current population and $|P|$ is the size of the population. The idea is that the lower value the worse window. We have two basic options: to select the window with the lowest value or to randomly select the window with the probability defined by this value. In the worst case, we keep removing windows during the diversification phase until the objective function is empty.

Another option would be to gradually increase a probability to remove a window. This way, it would remove the window only when it is necessary, that is during longer diversification phase. Another completely different approach, which



(a) Objective function (b) Description (c) Objective function and description in the same area of the image.

Figure 2.3: Visual representation of the objective function f and the description D in the same area.

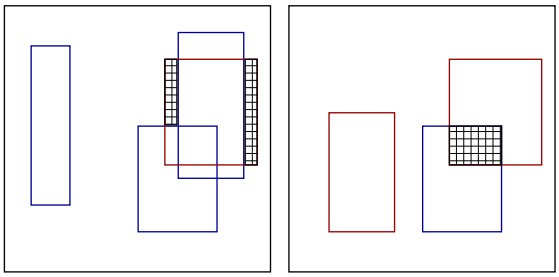
does not always simplify the objective function, would be to have a different inspiration criterion for the diversification phase, preferably more easier to be met.

2.4 Description metric for images

To build the idea of our definition, we start with a simple case. Let us assume, we have only one feature function F_{V_O}, the objective function f with one view V_O, the description $D = \bar{D}(I)$ of the individual (image) I with one view V_I, and their definition as in the figures 2.3a and 2.3b. To recall, V_O represents the optimized area by the optimization technique and V_I contains promising areas of the image I. For visual purposes, we can imagine both of them at the same area of the image, see figure 2.3c.

The general idea to define the extension $E(f, D)$ and the coverage $C(f, D)$ is based on areas of their windows. If D is added to f, the extension represents how much new area is gained by D. If the meaning of f and D is exchanged, that is, D is the new objective function, the coverage represents how much area is preserved from f. Nevertheless, we found more convenient to define the extension and the coverage from the point of view of a single window. These values are then useful to modify the objective function.

First of all, we need to raise several general questions. Do we want to prefer larger area? Does larger area represent more promising area? To recall, the windows were obtained from a "hypothetical" expert and we do not know why they are supposed to be promising. In the domain of images, we do not want to prefer larger areas. The larger does not necessary mean better. For instance, the area of the face in The Mona Lisa painting by Leonardo da Vinci is smaller than a huge simple circle in the whole painting. On the other hand, the larger area may have higher probability that the search finds something in it. However, ITS is based on splitting a problem to sub-problems. Otherwise, we could have only optimized the whole image. Do we know which is more promising window from two windows of the same size? To recall, the only information we obtained is that they are both promising. We assume that we do not know which area is more promising, that means the windows are incomparable. For instance, one



(a) Extension (b) Coverage

Figure 2.4: Visual definition of the area part of the extension and the coverage. Blue windows are from the objective function and red windows are from the description. The full objective function and description are shown in the figure 2.3.

of them might be already perfect and the second might be just at the beginning of the process, but the final result might surpass the first. Of course, we want the whole image to be "interesting" in the end. One could argue we could only optimize the sum area of the windows. Nevertheless, ITS tries to get there by the gradual incremental approach and not by the direct optimization where it could prematurely converge.

2.4.1 Extension

In the presented perspective, the extension $E(V_O, W_I)$ of the window $W_I \in V_I$ from the description D and the view V_O from the objective function f is how much area is gained by extending V_O with W_I. The figure 2.4a shows this idea. We compute the area of the complement between W_I and V_O and normalize it by the maximum possible gained area, which is the absolute complement of V_O. This represents a ratio of the gained space. We could have normalized it with the area of the view V_O. It would represent the relative improvement and not the global one. We also tried this approach but our proposed version produced visually better results in our preliminary experiments. The idea of our normalization is to get rid of the influence of the size of the area. However, the larger area can have a higher probability to more extend the rest of the image and we claimed we do not prefer larger areas. On the other hand, this preference is not explicit. To overcome this disadvantage, we consider only windows of size smaller than the quarter of the whole image.

Additionally, we can include information about the quality of the window W_I according to the feature F_{V_O}. To recall, the window W_I was obtained from a "hypothetical" supervisor. Nevertheless, it can be projected it to the view V_O associated with the feature F_{V_O}. The window with the higher value is a better candidate for extending the objective function. By putting these two information together, we define $E(V_O, W_I)$ as

$$E(V_O, W_I) = \frac{\text{area}(W_I \setminus V_O)}{\text{area}(V_O^{\complement})} \cdot F_{V_O}(W_I, I) \tag{2.6}$$

where $V_O^{\complement}$ is the absolute complement of V_O and *area* represents the size of the argument in pixels.

With more views in the objective function f, we compute the average extension of the window W_I as

$$E(f, W_I) = \frac{1}{\|f\|_V} \cdot \sum_{V_O \in f} E(V_O, W_I) \tag{2.7}$$

When the description D contains more than one window, we compute $E(f, D)$ as the average extension value for all windows by

$$E(f, D) = \frac{1}{\|D\|_W} \cdot \sum_{W_I \in D} E(f, W_I) \tag{2.8}$$

One could define it by the maximum value. It would represent the maximal extension of a single window from the description. When the description D contains no windows, we define the extension as

$$E(f, \emptyset) = 0 \tag{2.9}$$

because we gain nothing.

2.4.2 Coverage

In comparison with the definition of the extension, we look at the problem from the point of view of the objective function. In this perspective, the coverage $C(W_O, V_I)$ of the window $W_O \in V_O$ from the objective function f and the view V_I from the description D is how much area of W_O is preserved by defining the new objective function as V_I. The figure 2.4b shows this idea. We compute the area of the intersection between W_O and V_I and normalize it by the size of the window W_O. This represents a ratio of the preserved space. Similarly to the definition of the extension, it has the same problem. The larger area of V_I can have higher probability to preserve the window W_O or the smaller area of W_O can have higher probability to be preserved.

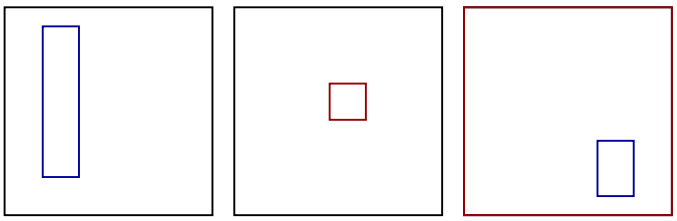
Again, we can include information about the quality of the window W_O according to the feature F_{V_O}. The window with the higher value is a better candidate for preserving. Finally, we define $C(W_O, V_I)$ as

$$C(W_O, V_I) = \frac{\text{area}(W_O \cap V_I)}{\text{area}(W_O)} \cdot F_{V_O}(W_O, I) \tag{2.10}$$

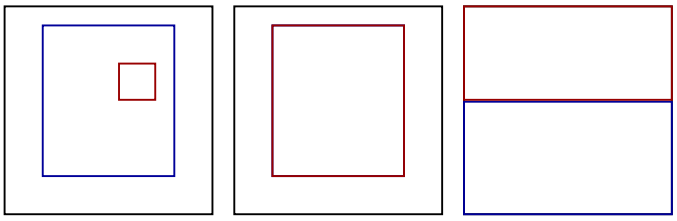
With more views in the description D, we compute the average coverage of the window W_O as

$$C(W_O, D) = \frac{1}{\|D\|_V} \cdot \sum_{V_I \in D} C(W_O, V_I) \tag{2.11}$$

Another option would be to virtually merge all views together. All windows from all views would be put in the same area and the window W_O would be checked against this "virtual" view. The idea is that we cannot compare the windows from experts, therefore we treat them all in the same way.



(a) $C = 0$, $E = 0$ (b) $C = 1$, $E \downarrow$ (c) $C = 1$, $E = 1$



(d) $C \downarrow$, $E = 0$ (e) Windows share the same area. $C = 1$, $E = 0$ (f) $C = 0$, $E = 1$

Figure 2.5: Simple examples of the definition of E and C. The objective function f is blue and the description D is red. A symbol $\downarrow$ represents a low value.

When the objective function f contains more than one window, we compute $C(f, D)$ as the average coverage value for all windows by

$$C(f, D) = \frac{1}{\|f\|_W} \cdot \sum_{W_O \in f} C(W_O, D) \tag{2.12}$$

Similarly to the definition of extension, one could define it by taking the lowest value. It would represent the minimal coverage of a single window in the objective function. The particular choice biases the resulting images and it highly depends on the definition of *inspire* method and the needs in a particular domain. When the objective function f contains no windows, we define the coverage as

$$C(\emptyset, D) = 1 \tag{2.13}$$

because we loose nothing.

2.4.3 Summary

It is worth noting that the extension E and the coverage C are not correlated, see the figure 2.5 for simple examples.

We use adaptive scaling for feature values, therefore the extension and the coverage are from the range of $[0, 1]$ most of the time. The value can only be bigger when the new bound has been found in the current generation.

By putting everything altogether, we define the extension $E(f, D)$ for the objective function f and the description D as

$$E(f, D) = \begin{cases} \frac{1}{\|D\|_W} \cdot \frac{1}{\|f\|_V} \cdot \sum_{W_I \in D} \sum_{V_O \in f} E(V_O, W_I) & \text{when } \|D\|_W \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

and the coverage $C(f, D)$ as

$$C(f, D) = \begin{cases} \frac{1}{\|f\|_W} \cdot \frac{1}{\|D\|_V} \cdot \sum_{W_O \in f} \sum_{V_I \in D} C(W_O, V_I) & \text{when } \|f\|_W \neq 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.15}$$

They are used for the description metric $\delta(f, D)$ computed as

$$\delta(f, D) = C(f, D)^\alpha \cdot E(f, D)^\beta \tag{2.16}$$

where $\alpha, \beta \in \mathbb{R}$ are parameters.

3. Experimental setting

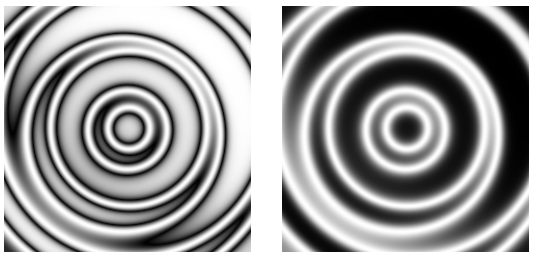
In this chapter, we describe settings for our experiments. Firstly, we introduce the representation of images. Then, we present the optimization techniques to create images with the concrete setting. Thirdly, we present used feature functions with typical examples of their produced images. Lastly, we describe the concrete settings of inspiration-triggered search for our experiments.

3.1 Image representation

An image is represented by a CPPN (see 1.1.1), which is a variation of an artificial neural network. We use the similar setting as in Picbreeder (see 1.5). Nevertheless, we use only feed-forward neural networks to describe images. It is worth noting that this image representation has infinite resolution. We choose to generate images with resolution of 256×256 pixels.

As described in 1.2.5, the feed-forward neural network to generate the image has 3 input nodes. X and Y coordinates in the image and the distance from the center of the image to the current pixel. These values are linearly scaled to the range of $[-1, 1]$. For instance, let us assume the image of size $N \times N$ pixels, X and Y coordinates from the domain $[0, N-1]$ and the distance from the center from the domain $[0, \frac{N-1}{\sqrt{2}}]$ are linearly scaled to $[-1, 1]$. The network has one output node representing the value of the current pixel transformed by an output function. We use the output function described in [89] defined as $255 \cdot |x|$, where x is the output value of the network. The brightness of pixel is darker the closer the output is to 0. Therefore, the output of 1 or -1 produces white. We also tried linear scaling as $[-1, 1] \rightarrow [0, 255]$, but the previous approach was found to produce the most interesting patterns in early generations. The images are sharper and it is harder for the network to produce black areas. The figure 3.1 shows a comparison.

The available CPPN activation functions are signed versions of sigmoid, Gaussian, and sine functions, resulting in a node output range of $[-1, 1]$, and linear function. The output range of the linear function is not restricted beforehand. Therefore, we explicitly limit the output value of the network to $[-1, 1]$. We



(a) $x \rightarrow 255 \cdot |x|$ (b) $[-1, 1] \rightarrow [0, 255]$

Figure 3.1: Comparison of an image generated with different output functions. The image (a) was generated by output function: $x \rightarrow 255 \cdot |x|$. The image (b) was generated by linear scaling of the output value: $[-1, 1] \rightarrow [0, 255]$.

use similar functions as in [85, 90] to capture regularities that appear frequently in nature (f.e. symmetry, repetition, and repetition with variation) without an intentional aesthetic bias.

Recurrent neural networks

We experimented with recurrent neural networks, where connections can form a directed cycle, and non-recurrent (feed-forward) networks in our preliminary experiments.

Using recurrent neural networks brings a question about how many times to active a network. To recall, the activation of recurrent network works differently in comparison with the feed-forward network. In feed-forward networks, a single activation of the network is done a layer by layer. It does not contain cycles, therefore we get immediately the output. On the contrary, the recurrent network contains directed cycles and it must be repeatedly activated. In one activation, each node takes its inputs and computes its output. This yields the problem of how many times to activate the network.

Firstly, the recurrent network does not have to converge or it can converges in cycles, see the figure 3.2 for an example. Furthermore, the problem whether the network converges or not is undecidable problem [86]. Recurrent networks are computationally equivalent to Turing machines and this problem represents the halting problem for Turing machines.

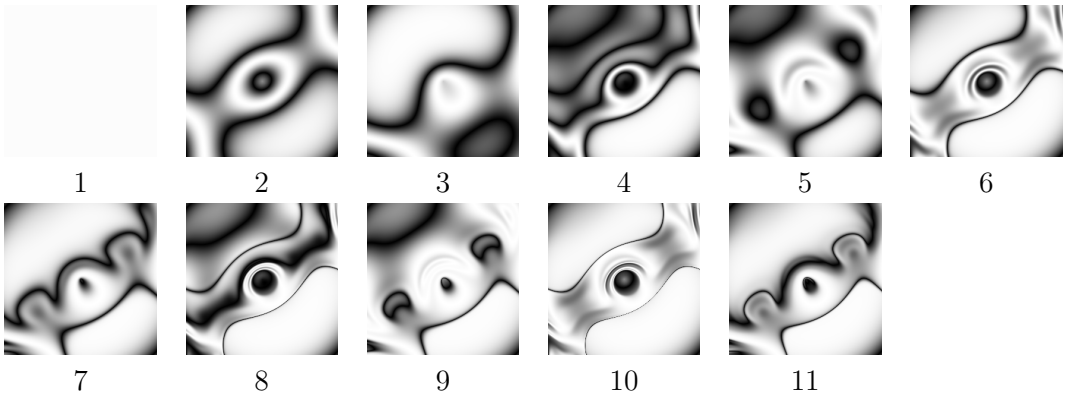


Figure 3.2: Example of a convergence of a recurrent neural network in cycles. Numbers represent the number of activations of the recurrent neural network. Afterwards, the network repeats last four images $(8 - 11)$ in cycles.

Secondly, let us assume, we want to at least propagate all node values to the output node. The minimal number of activations is the length of the longest simple path in the network, that does not have any repeated vertices, from any input to any output of the network. Nevertheless, the longest path problem is NP-hard problem.

Finally, we use neural networks to generate images and not to describe a behaviour over time with a memory, for instance, such as in a robotics domain. Also, we want the image to be described by a simple formula. With the given number of activations, we can unfold the recurrent network to feed-forward network over time. From this point of view, the recurrent link creates easily more

nodes. Thus, a mutation of adding a recurrent link in NEAT (see 1.2.2) results in adding more nodes in the unfolded feed-forward network.

Our preliminary experiments showed the similar performance for both types of networks. Therefore, we chose only to evolve feed-forward neural networks to describe images by simple formulas. On the other hand, Picbreeder uses topology of any type.

3.2 Underlying optimization technique

We chose NEAT (see 1.2.2) as our underlying optimization technique. Its incremental complexification, that is incrementally adding new nodes over generations, is similar to our approach. ITS starts with simple images and makes them more complex over time. Picbreeder also uses NEAT. Our implementation is based on MultiNEAT [13] which is a portable software library implementing NEAT.

We use the similar setting as in the original NEAT, Picbreeder, and similar experiments [94, 24]. The table 3.1 shows NEAT parameters used in our experiments. These parameters were chosen based on our preliminary experiments, in which we optimized a set of features. We did not try to optimize the setting towards, that one feature performs best or all of them together. Furthermore, no free lunch theorems [103] tell us that in general it is not even possible to optimize settings such that each feature separately performs best.

We use population of 150 individuals with a number of species between 10 and 15, which is also used in HyperNEAT algorithm to evolve CPPNs describing connectivity patterns of neural networks. Offspring has 0.05 probability of adding a node, 0.04 probability of adding and removing a link, 0.001 chance of removing a node, 0.9 chance of link weight mutation, and 0.02 chance of changing an activation function for a node. Recurrent connections within the CPPN are not enabled [1]. The disjoint and excess node coefficients are both 2.0 and the weight difference coefficient is 1.0. The activation function difference coefficient is set to 1.0 allowing to distinguish networks with different activations functions. The compatibility threshold is 6.0, the compatibility modifier is 0.3, and the minimal compatibility threshold is 0.2. The weight range of CPPN is $[-8, 8]$, the maximum perturbation for a weight is 1.0, and the maximum magnitude of a replaced weight is 2.0.

The available CPPN activation functions are signed versions of sigmoid, Gaussian, and sine functions, and linear function, all with equal probability of being added. The initial population of NEAT starts with Gaussian function as the output's activation function, but this function can be later changed by mutations.

3.3 Features

In this section, we present features we use for our experiments. Our domain is mainly used in evolutionary art (see 1.2.4), therefore we took an inspiration from that field. We tested measures from the field of computational aesthetics (see 1.3)

[1] We fixed a bug in the implementation of MultiNEAT that created connections between nodes on the same layer even with recurrent connections turned off. Thus, it always created recurrent neural networks.

Parameter	Value	Parameter	Value
PopulationSize	150	MutateRemSimpleNeuronProb	0.001
DynamicCompatibility	True	RecurrentProb	0.0
MinSpecies	10	MutateWeightsProb	0.9
MaxSpecies	15	MutateWeightsSevereProb	0.5
YoungAgeTreshold	15	WeightMutationRate	0.75
YoungAgeFitnessBoost	1.1	WeightMutationMaxPower	1.0
SpeciesMaxStagnation	15	WeightReplacementMaxPower	2.0
OldAgeTreshold	35	MaxWeight	8.0
SurvivalRate	0.2	MutateNeuronActivationTypeProb	0.02
CrossoverRate	0.75	DisjointCoeff	2.0
OverallMutationRate	0.25	ExcessCoeff	2.0
InterspeciesCrossoverRate	0.01	WeightDiffCoeff	1.0
MultipointCrossoverRate	0.6	ActivationFunctionDiffCoeff	1.0
RouletteWheelSelection	False	CompatTreshold	6.0
MutateAddNeuronProb	0.05	MinCompatTreshold	0.2
MutateAddLinkProb	0.04	CompatTresholdModifier	0.3
MutateRemLinkProb	0.04		

Table 3.1: The NEAT parameters used in our experiments.

and measures defining a sharpness of the image (see 1.4). We subjectively chose measures that produced some visual results. For instance, information entropy (see 1.3.2), and Benford law (see 1.3.3) mainly produced images with simple gradients or an area with one colour. Thus, optimization by NEAT for them converged in a few generations. We suggest they are more appropriate for fine art than for our artificially created images. Our implementation uses OpenCV library [11] to manipulate images.

3.3.1 Global contrast factor

In comparison with the original version described in 1.3.4, we only use super-pixels of sizes up to 50. In the original paper, the authors used images with resolution of 800×600. Our images have smaller resolution of 256×256. Thus, we use super-pixels of following sizes: 1 (the original image), 2, 4, 8, 16, 25, and 50, and we re-define weighting factors w_i (see formula 1.11) as

$$w_i = (-0.406385 \cdot \frac{i}{7} + 0.334573) \cdot \frac{i}{7} + 0.0877526 \tag{3.1}$$

where $i \in \{1, \ldots, 7\}$. The measure M_{GCF} for an image I is re-defined (see formula 1.10) as

$$M_{GCF}(I) = \sum_{i=1}^{7} (w_i \cdot C_i) \tag{3.2}$$

Examples of images created by the optimization of global contract factor are shown in the figure 3.3.

Figure 3.3: Examples of images created by optimization of global contrast factor.

3.3.2 Relaxed symmetry

Examples of images created by the optimization of relaxed symmetry described in 1.3.5 are shown in the figure 3.4. We find these images visually very simple.

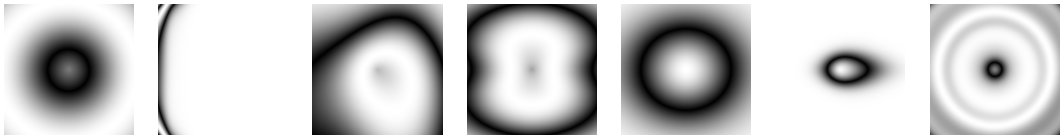


Figure 3.4: Examples of images created by optimization of relaxed symmetry.

3.3.3 Normalized variance

Examples of images created by the optimization of normalized variance described in 1.4.2 are shown in the figure 3.5. Similarly to relaxed symmetry, the images are visually very simple.

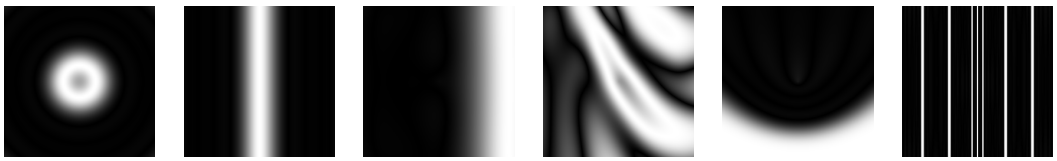


Figure 3.5: Examples of images created by optimization of normalized variance.

3.3.4 Tenengrad

Examples of images created by the optimization of Tenengrad described in 1.4.1 are shown in the figure 3.6. The images are similar to images created by the optimization of global contract factor.

Figure 3.6: Examples of images created by optimization of Tenegrad.

3.3.5 Choppiness

Choppiness measure M_C is computed as the average standard deviation of pixels over all 5×5 windows within the image. The same measure was used in [46].

Examples of images created by the optimization of choppiness are shown in the figure 3.7. The images mainly contain "noise" or more precisely high frequencies.

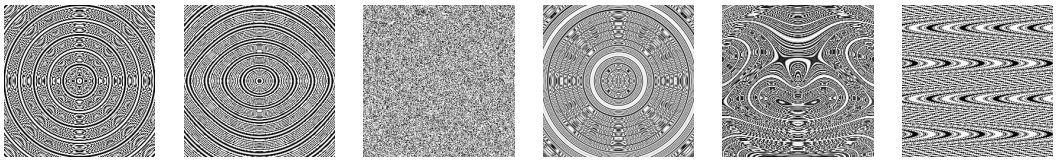


Figure 3.7: Examples of images created by optimization of choppiness.

3.3.6 Image complexity by JPEG compression

This measure uses the image complexity defined by Machado and Cardoso described in 1.3.1. We compress an image by JPEG compression with a quality setting of 75%. The measure M_{IC} for the image I is defined as

$$M_{IC}(I) = \frac{RMS(I)}{CompressionRatio(I)} \tag{3.3}$$

where RMS is the difference between the original and the compressed image expressed as the root mean square. The compression ratio is the ratio between the original and the compressed image size.

Examples of images created by the optimization of image complexity by JPEG compression are shown in the figure 3.8. Similarly to choppiness, the images mainly contain high frequencies.

Figure 3.8: Example of images created by optimization of image complexity by JPEG compression.

3.3.7 Maximum of absolute Laplacian

We define another simple way to estimate a sharpness of an image. We take the maximal value from absolute values of Laplacian of an image. The measure M_{MAL} of the image I is defined as

$$M_{MAL}(I) = \max(|L(I)|) \tag{3.4}$$

where $L(I)$ is computed by convolving the image I with the Laplacian operator

$$L(I) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} * I \tag{3.5}$$

where $*$ denotes the 2-dimensional convolution operation.

Examples of images created by the optimization of maximum of absolute Laplacian are shown in the figure 3.9. The optimization does not produce images with high frequencies in comparison with Tenengrad. The features is basically an edge detector, therefore the images are sharp.

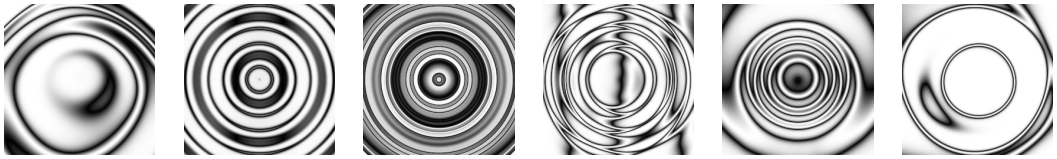


Figure 3.9: Examples of images created by optimization of maximum of absolute Laplacian.

3.3.8 Feature as penalty

For later purposes, we also need to define a feature (penalty) to penalize high frequencies. A simple solution is to use choppiness and image complexity by JPEG compression, see their produced images in the figure 3.7 and 3.8.

Penalty by choppiness

We use choppiness M_C described in 3.3.5 and define a penalty P_C for an image I as P

$$P_C(I) = \begin{cases} \dfrac{1}{M_c(I)} & \text{when } M_C \geqq 60 \\ 1 & \text{otherwise} \end{cases} \tag{3.6}$$

This definition is based on our preliminary experiments.

Penalty by image complexity by JPEG compression

We use image complexity by JPEG compression M_{IC} described in 3.3.6 and define a penalty P_{IC} for an image I as

$$P_{IC}(I) = \begin{cases} \dfrac{1}{M_{IC}(I)} & \text{when } M_{IC} > 1 \\ 1 & \text{otherwise} \end{cases} \tag{3.7}$$

We only want to penalize high frequencies and not to support "simple" images, thus the condition $M_{IC} > 1$ is used.

3.4 Inspiration-triggered search

In this section, we describe the shared setting of ITS for our experiments. Experiments differ only by the inspiration threshold I (used by the inspiration criterion), the threshold R to remove a window from the objective function (used by the operator to modify the objective function), and a set of features.

We run ITS for 40 creative cycles. Without a convergence of the optimization, it would be the optimization by NEAT for 400 generations and at least 20 possible times to change the objective function. Nevertheless, each trial is unique. The number of iterations for the underlying optimization is 10 and the convergence limit for the optimization is 10.

Modification of the objective function

The description to modify the objective function is selected by the best description metric value. The modification of objective function is achieved by a set of operators. We typically use one or two operators: adding or removing a window. To add a new window to the objective function, we use the random variant in which the probability to select the window is defined by the extension value. We also tested the other variant of the selection by the best extension value. The random version produced visually better images in our preliminary experiments. If we also want to remove a window from the objective function, we will set it as the first operator. This way, first of all, one window is removed from the objective function and then the new window is added by the second operator. The operators use the extension and the coverage values defined by the original objective function. For simplification, when the operator to remove a window from the objective objective is not used, we say that $R = 0$.

After the change of the objective function, we reset the current information about the best fitness value and the number of stagnations in NEAT. This way, the objective function is not penalized by mistakes from the previous objective function. Additionally, the reset of the best fitness value found so far is crucial. Objective functions may differ in the number of their windows and the upper bound for the fitness value in NEAT with our adaptive scaling is the number of windows. Without the reset, the new objective function with less windows could immediately converges.

Description

We use only one view in the description of an image. We define windows by detection of contours in the image. As mentioned in 2.3.1, this should represent a similar behaviour as in visual areas of human brains [6]. As a side note, we also used a detection of human faces in our preliminary experiments.

Firstly, we detect edges by the Canny Edge detector and the result is used to find contours in. We use functionality from OpenCV library. The window for the view is defined by a bounding rectangle of a contour. We consider only windows with an area bigger than 2000 and smaller than the quarter size of the whole image. It is used in order to overcome possible disadvantages of large windows (see 2.4.1). Figure 3.10 shows several examples of descriptions.

Diversification method

To diversify the population, we use mutations from NEAT. Other options would have been to use another form of random walk or novelty search. We iterate through the whole population and we apply mutations from the current instance of NEAT used for the optimization to each individual. We apply it for two times to get a more diverse population. This operation can harm an arrangement of individual into species in NEAT. Therefore, we forget the current speciation and we again separate the individuals into new species. This way, the diversification can change the number of species in NEAT.

During the diversification phase, if the inspiration criterion is not met with the objective function having no windows in 10 tries, NEAT will be reset. In

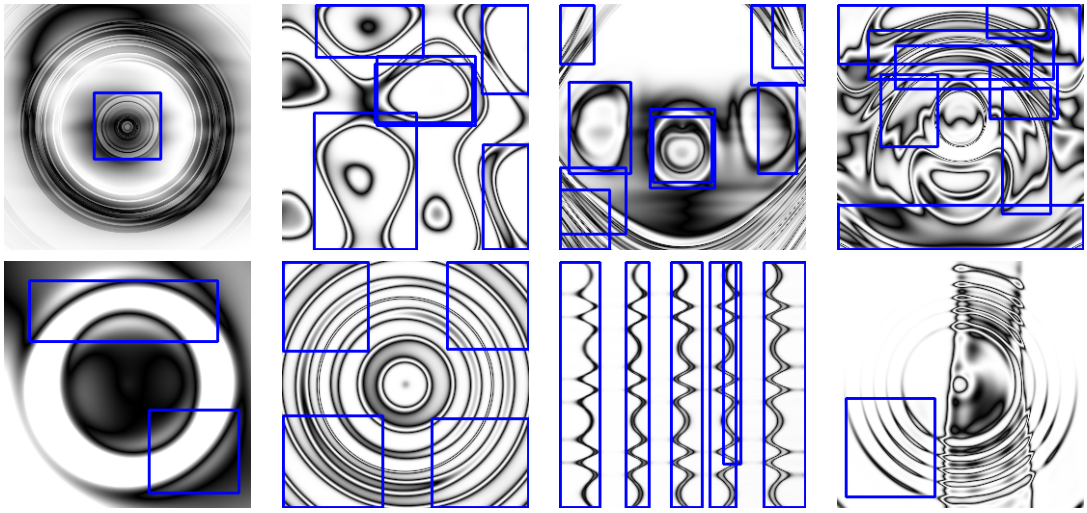


Figure 3.10: Examples of views by detection of contours.

other words, ITS starts again from scratch because it is at the same situation.

Extension

The extension values of description metric have much smaller values in comparison with the coverage values. One approach to increase their values is to change α and β parameters of δ. We take another approach and directly scale the area of the complement (see 2.4.1) by the function

$$f(x) = \frac{2x}{x + \omega} \tag{3.8}$$

where ω is a constant representing the maximum value of x. Thus, we re-define $E(V_O, W_I)$ from the original formula 2.6 to

$$E(V_O, W_I) = \frac{2 \cdot \text{area}(W_I \setminus V_O)}{\text{area}(W_I \setminus V_O) + \text{area}(V_O^{\complement})} \cdot F_{V_O}(W_I, I) \tag{3.9}$$

This decision was particularly made for our size of images and with simple settings of $\alpha = 1$ and $\beta = 1$.

4. Evaluation

In this chapter, we evaluate proposed inspiration-triggered search (ITS) for our testing domain of images. Firstly, we introduce two methods for a comparison. Secondly, we describe several selected runs of ITS to get more familiarized with its behaviour. Thirdly, we compare ITS with different settings and with the underlying optimization using the same set of feature functions. Lastly, we provide a summary of the results from the comparison.

4.1 Methods for comparing complexities

Our goal is to compare ITS with the underlying optimization technique using the same set of feature functions and to show whether ITS produces "more complex" and visually better images or not. From now on, we will call the underlying optimization technique an "uninspired search."

In the domain of images, we can visually compare the generated images. Nevertheless, this comparison is subjective. We would also like to have a non-subjective measure. Especially, we would like to compare images by their "complexities." However, how to define a complex image is a difficult problem by itself.

4.1.1 Complexity test by optimization

Our initial idea is to test whether the image can be found or not by the used optimization technique using the image as an objective function. We define the objective function as Euclidean distance to the desired image. In other words, it is the distance in the pixel space between two images and the optimization technique minimizes this pixel distance. We can define that the image is complex when it cannot be found by this approach. We split the domain of pixel distances into two ranges for the classification purposes based on our preliminary experiments, in which the values were usually scattered into two clusters, see the figure 4.1. From these results, we defined a border for the classification. When the pixel distance is lower than 16000, the image is classified as simple, otherwise it is labelled as complex image. We found from our preliminary experiments that 600 generations is sufficient for this complexity test. When it did not converge before 600 generations, best pixel distance improved for less than 1000 for another 400 generations and this improvement was insignificant in comparison with the distance values.

For a single image, we gather the pixel distances of the best individuals from the last generations of all trials. This represents the difficulty to found the image. To compare an algorithm, we select representative generated images from it and we aggregate their results together. For instance, the figure 4.1 shows a comparison by boxplots for uninspired search using one feature function.

However, our experiments revealed that the optimization technique is only able to find visually simple images. For instance, images in the figure 4.3 at coordinates $[2, 4]$, $[2, 5]$, $[4, 1]$ (row, column) were able to be found, but visually simple images at coordinates $[1, 5]$, $[2, 2]$, $[3, 7]$ were not. The complexity test by

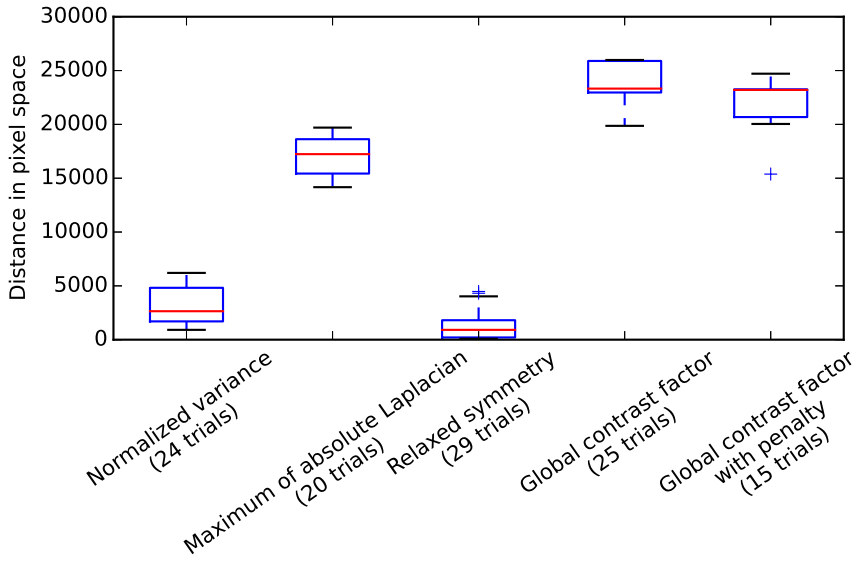


Figure 4.1: The results of the complexity test by optimization for generated images from uninspired search using one feature function.

optimization labels the latter as complex. However, we would like to label all these images as simple according to our subjective perspective. Furthermore, we consider all images used for the complexity test in the figure 4.1 as simple.

Our explanation of this problem is based on the idea shown in the figure 4.2. We split the domain of images into four classes. The first class (I) represents images containing simple gradients, and only one colour, for instance, images at coordinates $[2, 5]$, $[5, 3]$ (row, column) in the figure 4.3. The second class (II) contains images with simple shapes, patterns, and regularities that are easily created due to the used image representation by CPPN, for instance, images at coordinates $[1, 5]$, $[2, 6]$, $[5, 6]$. The third class (III) contains the "true complex" images, that we would like to create. The fourth class (IV) contains "chaotic" images with high spatial frequencies, for instance, images at $[1, 2]$, $[2, 1]$, $[4, 2]$, $[5, 5]$. The similar idea was used for cellular automata in a work [4] in which complexity classes in computation refer to the time it takes for a halting computation to complete with respect to the size of its input.

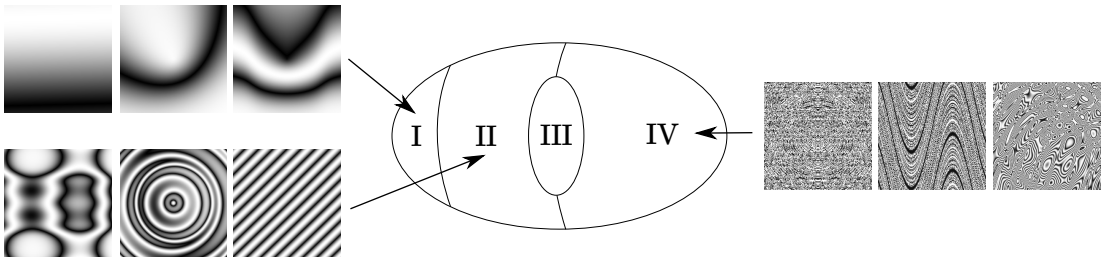


Figure 4.2: Visual idea of splitting the domain of images into four classes according to their complexities. Class I contains images with simple gradients, and only one colour. Class II contains images with patterns, and regularities. Class III represents "true complex" images. Class IV contains images with high spatial frequencies.

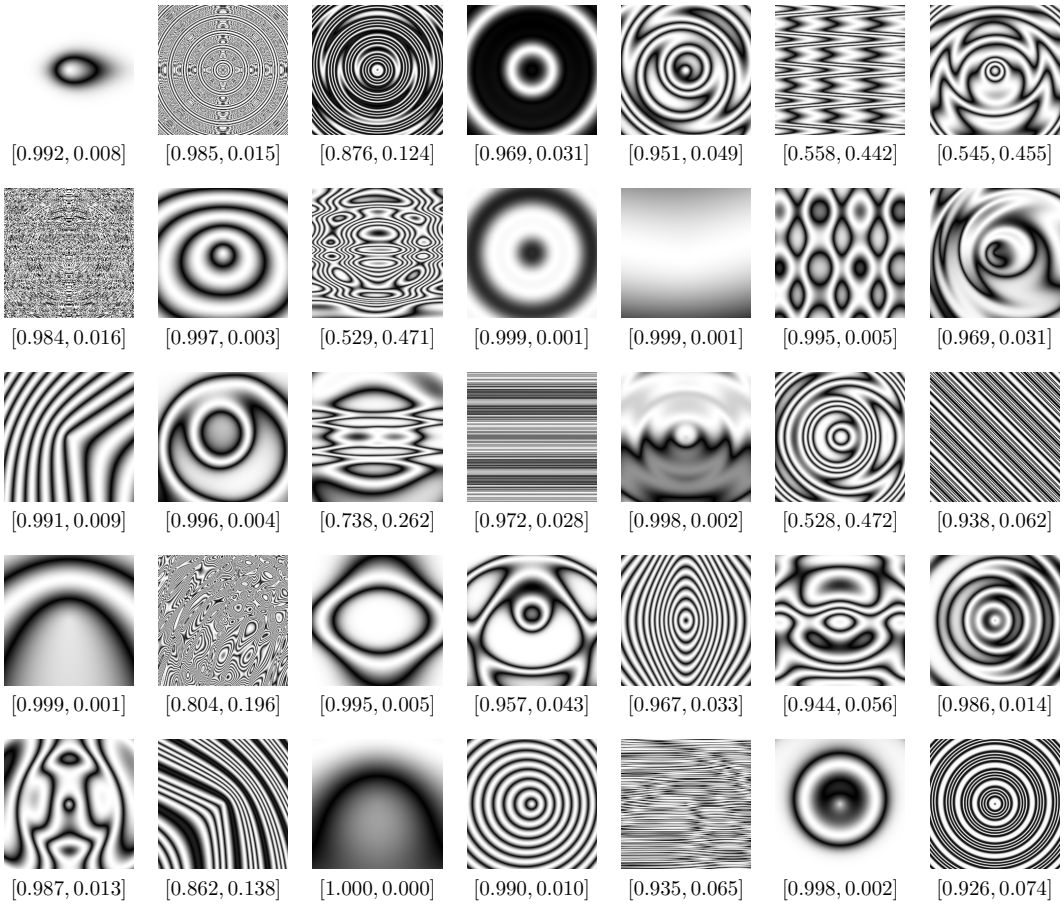


Figure 4.3: Examples of simple images from the data set to train a complexity classifier with their final complexity classifications. The classification of the image consists of the simplicity and the complexity node.

We believe that the complexity test by optimization is only able to distinguish the first class (I) from the rest (II, III, and IV). For instance, we think that images used for the complexity test in the figure 4.1 from normalized variance and relaxed symmetry belong to the class I, from maximum of absolute Laplacian to the class II, and global contrast factor to the class II and IV. However, we would only like to label an image as complex when it is from the class III.

Our first approach to solve this problem is to penalize images with high spatial frequencies. The idea is that we do not want to create images from the class IV. Therefore, a fitness value of an individual is multiplied by two penalties: choppiness and image complexity by JPEG (see 3.3.8). Unfortunately, it did not help too much. The images from global contrast factor with this penalty used for the complexity test are still classified as complex, see the figure 4.1. Nevertheless, we will always use this penalty for feature functions creating images from the class IV.

4.1.2 Complexity classifier

Due to the previously mentioned problems, we tried to distinguish these classes by machine learning methods. Firstly, we created a data set to distinguish simple

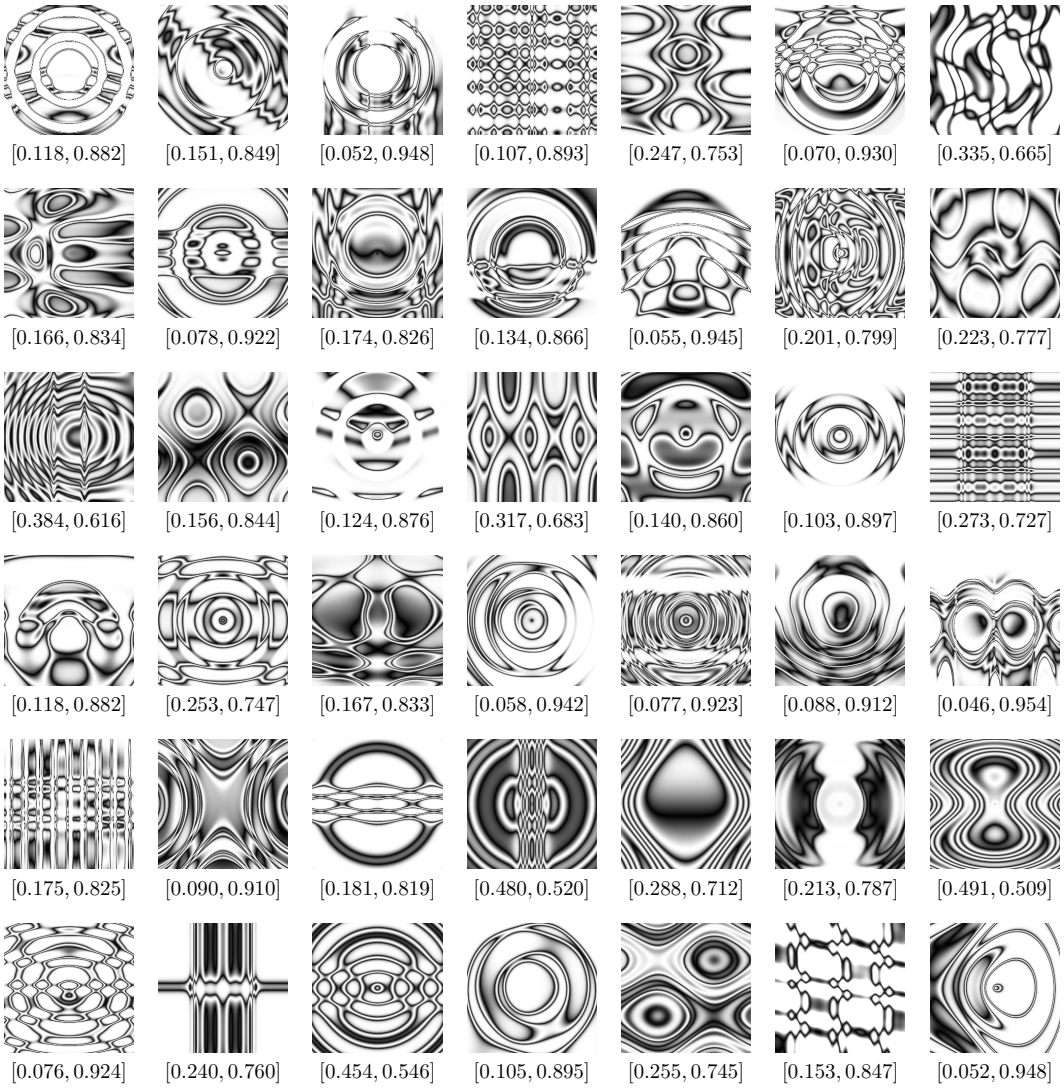


Figure 4.4: Examples of complex images from the data set to train a complexity classifier with their final complexity classifications. The classification of the image consists of the simplicity and the complexity node.

and complex images from our preliminary experiments. Some of these images are from the direct optimization using different sets of features. Others are from different versions of algorithms, that led to the proposed algorithm, with different settings and using different sets of features. Therefore, the images are not biased by our proposed algorithm. The images only share the same representation by CPPN. We mainly labelled an image as simple when it contains simple gradient, simple and blurred objects (class I and II), high spacial frequencies (class IV), and when it can easily be created by a random mutation during initial populations in NEAT (class II), see the figure 4.3 for examples. For the latter, we used the knowledge from our preliminary experiments. We labelled an image as complex when it is sharp or it includes more objects, see the figure 4.4 for examples. The final data set consists of 4113 simple and 1343 complex images. Additionally, we created another unlabelled data set of 113 images. We used it to visually compare the results of the classification and to interpret behaviours of different classifiers.

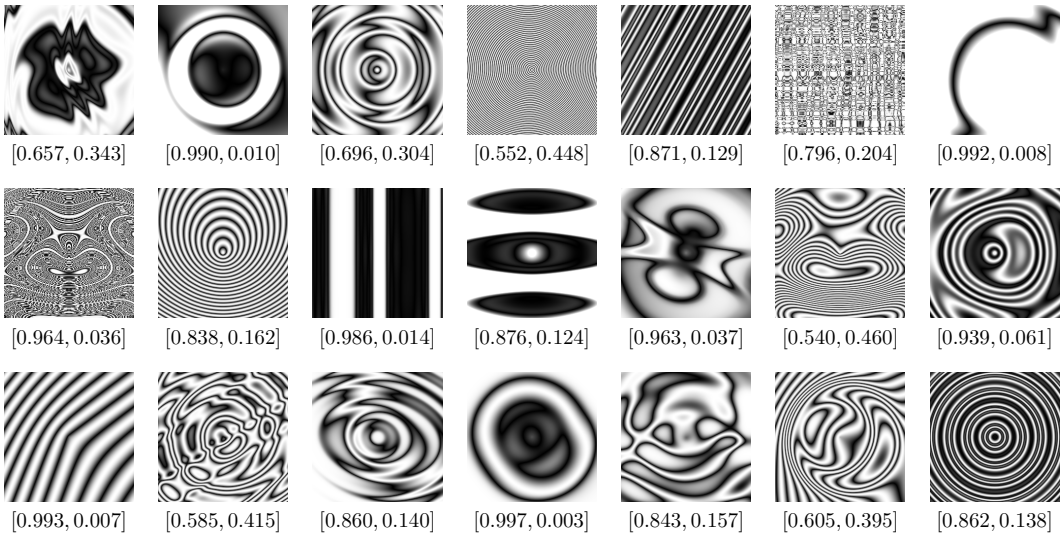


Figure 4.5: Examples of images classified as simple by the complexity classifier. The classification of the image consists of the simplicity and the complexity node.

Secondly, we chose 7 features to describe an image: normalized variance, maximum of absolute Laplacian, Tenengrad, choppiness, relaxed symmetry, global contrast factor, and image complexity by JPEG compression. This set is not biased by our proposed algorithm, because we never use all of them together. For instance, maximum of absolute Laplacian was even left out from our final experiments. It is basically an edge detection and sharp images are classified as complex. Thus, the optimization of this feature always produced complex images according to the final classifier. Another option would have been to use the whole image or its subset and for instance to use deep learning techniques. We randomly split the data set into training and testing data set with 25% of data in the testing set. We used the whole data set to obtain the mean and the standard deviation for each feature. It is used to scale the input values for a classifier to be centred with the unit variance.

We tested neural networks with a different number of nodes in one hidden layer and support vector machines (SVM) with different kernels. For both of them, we use libraries for Python: PyBrain [82] and Scikit-learn [67]. We trained neural networks by back-propagation for 50 epochs. The neural networks have two output nodes and a bias. The first output node classifies the image as simple and the second output node as complex. We will call them a simplicity node and a complexity node. The node with the higher value wins the classification. We did not want to only choose the best performing classifier. We subjectively tested the results on the unlabelled data set. We rather chose a classifier that tends to classify images as simple than the other way around. We chose the neural network with 10 nodes in one hidden layer. It had 91.5% classification accuracy on the testing data. For instance, the neural network with 14 hidden nodes had 92.08%, SVM with polynomial kernel of degree 3 had 91.94%, and SVM with radial basis function kernel had 92.23% classification accuracy. Nevertheless, they tended to classify more images as complex, which we did not want to.

This way, we are able distinguish simple and complex images to some extent. The main problem is that sometimes we are not even able to tell whether the

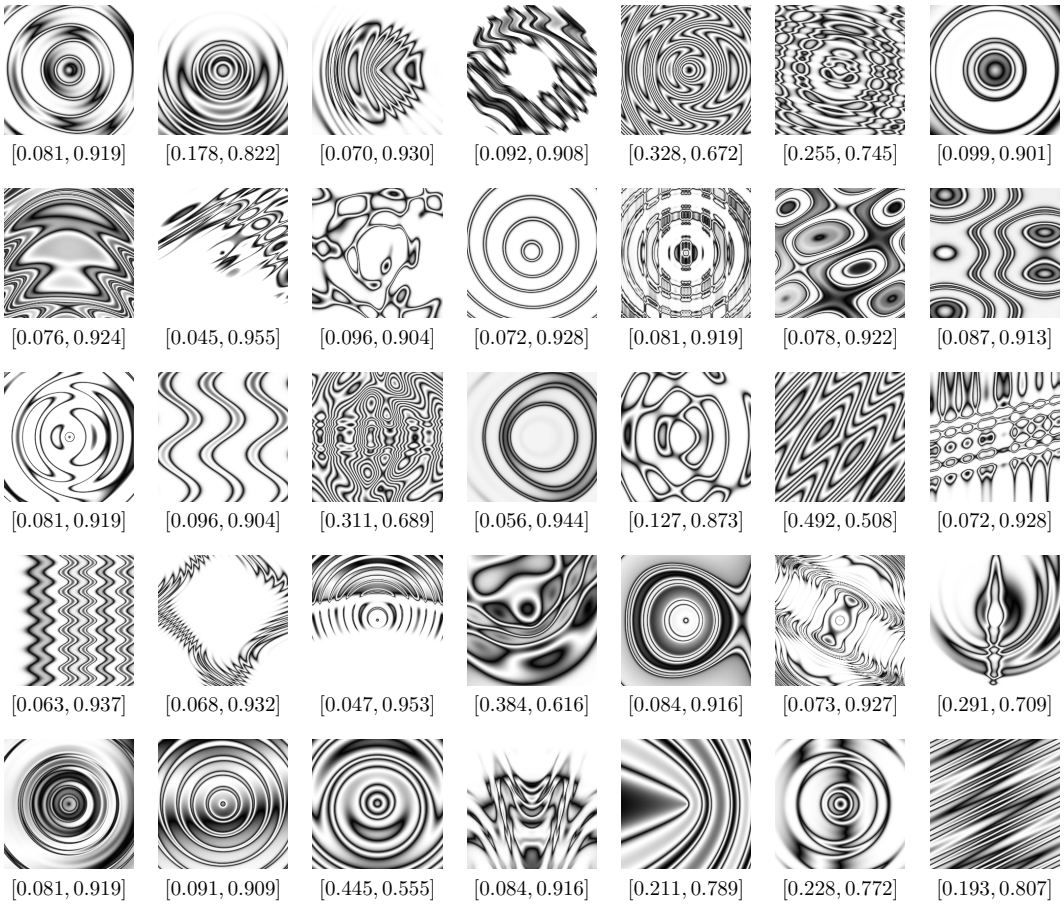


Figure 4.6: Examples of images classified as complex by the complexity classifier. The classification of the image consists of the simplicity and the complexity node.

image is complex or not. Figures 4.5 and 4.6 show examples of the final classification. We are satisfied that images classified as simple are not "complex" according to our subjective perspective. However, we consider several images classified as complex as "simple" (or more precisely, false positives), for instance, images at coordinates $[2, 4]$, $[5, 6]$ (row, column) in the figure 4.6. Images at coordinates $[3, 6]$, $[5, 3]$, and $[1, 5]$ are at the border of the classification and we would rather label them as simple images.

4.2 Visualization of ITS

We describe one run of ITS with one feature for which we can visually show the changes of the objective function. It used relaxed symmetry as the feature function, $I = 0.02$ as the inspiration threshold, and $R = 0.4$ as the removal threshold to remove a window from the objective function.

We split the run of ITS into evaluations by computationally expensive operations of the population. The optimization by NEAT and the testing of the inspiration criterion, in which the description for each individual is created, are the most computationally expensive operations. The diversification of the population and the modification of the objective function are computationally trivial in comparison with the rest. Therefore, we cluster them together as shown in the

figure 4.7. Each column represents one evaluation. For this reason, we cannot straightforwardly compare ITS with the underlying optimization technique. One generation of the NEAT is also one evaluation. However, ITS has more different phases and NEAT contains only the optimization operation.

Firstly, we describe the behaviour shown in the figure 4.7. ITS started with the diversification phase. We consider this evaluation as the initial phase creating a random population, thus we label it as the 0^{th} evaluation. In the 1^{st} evaluation, ITS tested the inspiration criterion and the condition was not met because another diversification phase followed. The best description metric value was 0.00098, which is lower than the threshold $I = 0.02$. In the 2^{nd} evaluation, the inspiration criterion was met. The graph 4.9f shows that the best description metric value was 0.05. The initial objective function with one window was defined. From the 3^{rd} evaluation, the optimization using this objective function started. It continued for 10 evaluations (or generations in terms of the optimization technique). Then, the limit to optimize, which is set to 10, was reached. The inspiration criterion was not met in the following 13^{th} evaluation and ITS started again to optimize from the next evaluation. This behaviour repeated until the 52^{nd} evaluation, in which the optimization reached the limit of stagnations, which is set to 10. In other words, the best fitness value in NEAT had not improved for the last 10 generations. The graph 4.9f shows that the best description metric values were always lower than the threshold I. Due to the convergence, ITS tested the inspiration criterion to find out whether the objective function can be changed without the diversification of the population or not. The condition was satisfied, therefore the objective function was modified and ITS continued to optimize from the next evaluation. Later in the 63^{rd} and 85^{th} evaluation, one window was removed from the objective function while its modification. The graph 4.9h shows that even the best convergence values were below the threshold $R = 0.4$, therefore the operator to modify the objective function was able to remove one window. The convergence limit was again reached in 99^{th} evaluation. The inspiration criterion was not met this time, therefore the population was diversified. In the following 100^{th} evaluation, the condition was not again satisfied, therefore the objective function was simplified by removing one window, and the population was diversified. The same situation happened again in the next evaluation and the last window from the objective function was removed. This situation can be seen in the graph 4.9e, where the total number of windows is 0. In the following 102^{nd} evaluation, the condition was met and the objective function with one window was defined. The rest of the run shown in the figure 4.8 should be clear.

Secondly, we can analyse the run by the complexity classifier. The graph 4.9a shows increasing values of the complexity node and the graph 4.9b shows decreasing values of the simplicity node. Only the complexity node values do not tell the whole story. For instance, the complexity node value may be 0.8, which seems as a complex image. However, the simplicity node value may be 0.9, therefore the image is classified as simple. For this reason, we use classification difference defined as complex node value minus simplicity node value, see the graph 4.9c. Images are classified as complex, when their classification difference values are above zero, and vice versa. The graph 4.9d shows the ratio of images classified as complex in the population. This graph uses only the discrete results of the classification. The ratio rose up to 0.9, nevertheless the classification difference

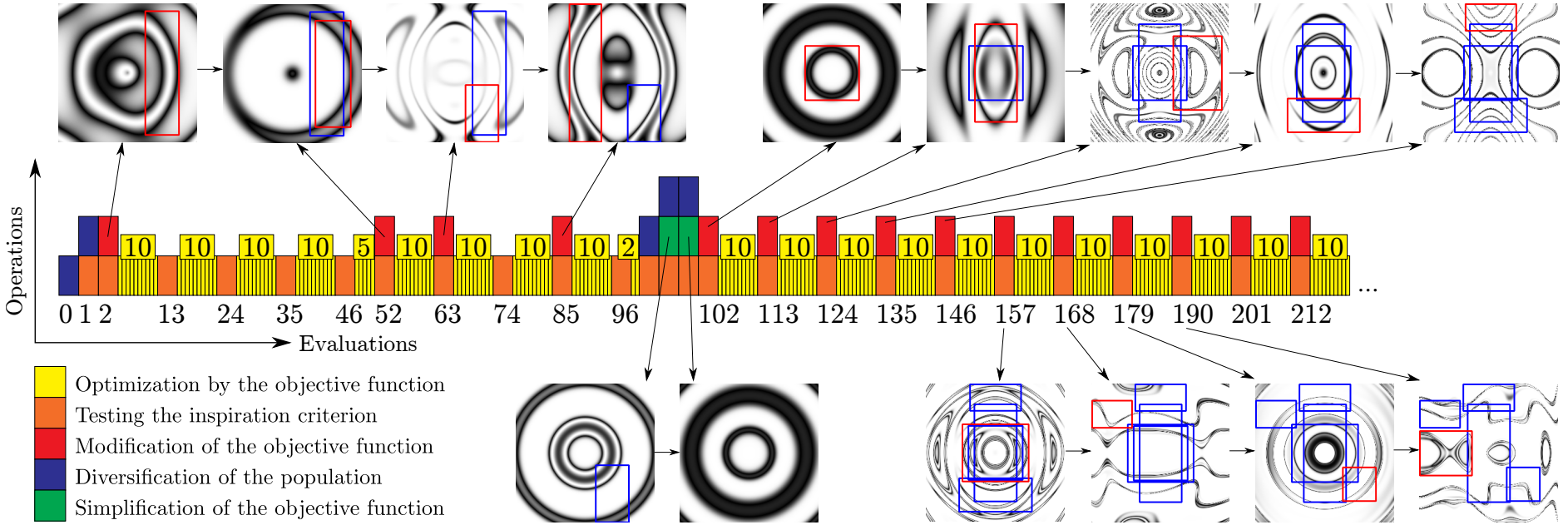


Figure 4.7: Visualization of the selected run of ITS using one feature shows used operations over evaluations. Each rectangle represents one particular operation. Rectangles of the optimization are grouped together and they use smaller rectangles with their total number. The objective functions are shown for each change until the 200$^{\text{th}}$ evaluation. A red window is the new added one into the objective function and blue windows are from the previous objective function. For the modification of the objective function, the shown image was used to modify the objective function, or more precisely, the description of the shown image. For the simplification of the objective function, the shown image had the highest description metric value in the population. The rest of the run is shown in the figure 4.8.

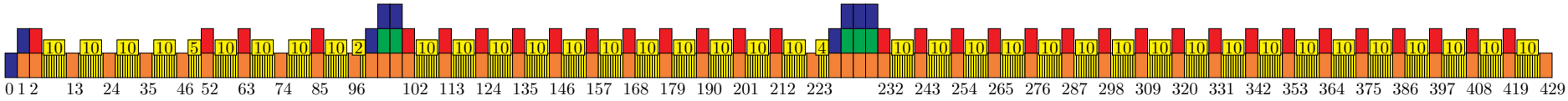
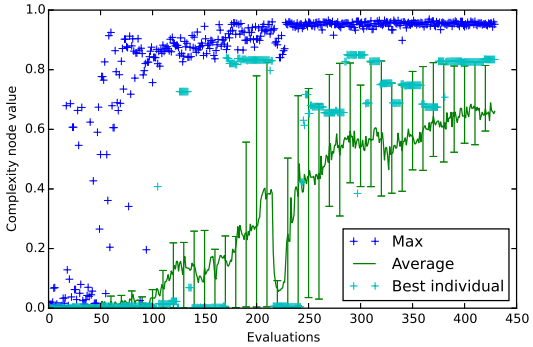
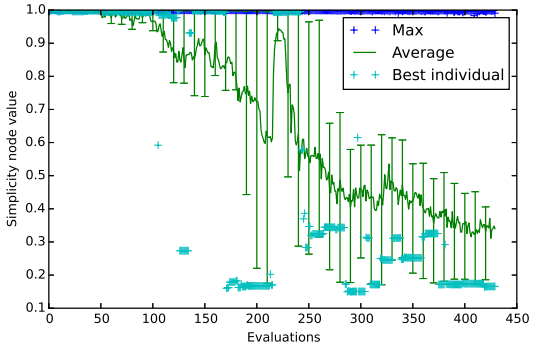


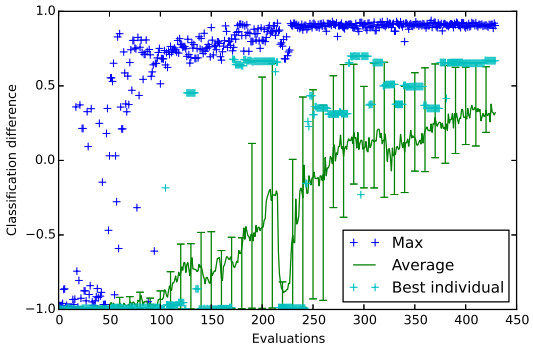
Figure 4.8: Operations of the selected run over evaluations. Concrete description is given in the figure 4.7.



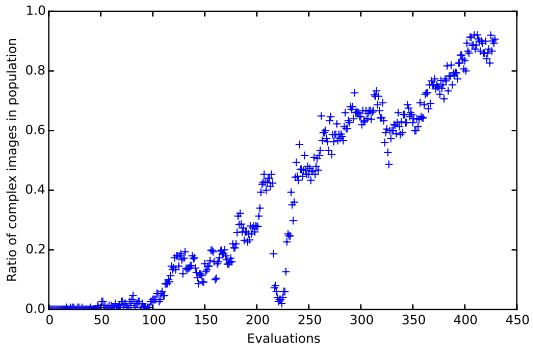
(a) Complexity node values



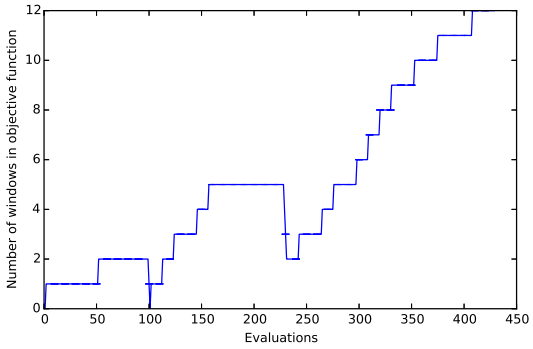
(b) Simplicity node values



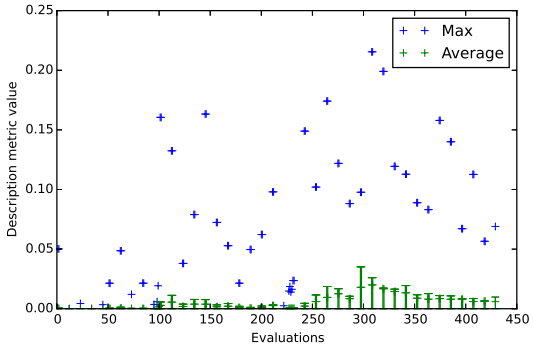
(c) Classification difference values



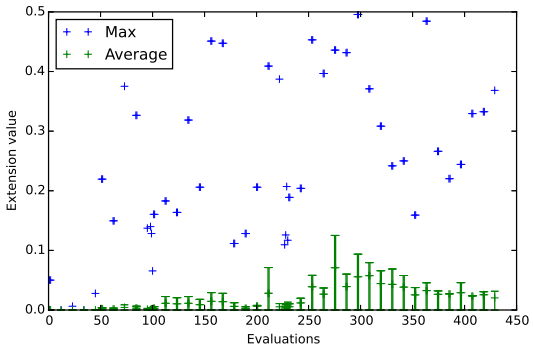
(d) Ratio of images classified as complex



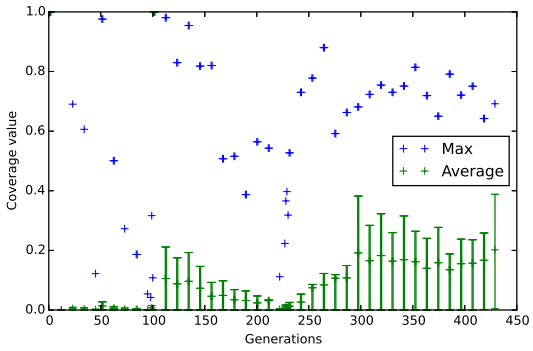
(e) Number of windows in the objective function



(f) Description metric values



(g) Extension values



(h) Coverage values

Figure 4.9: Information about the selected run of ITS over evaluations. Error bars have the same meaning as in boxplots, that is, the bottom and the top are the 25^{th} and the 75^{th} percentiles.

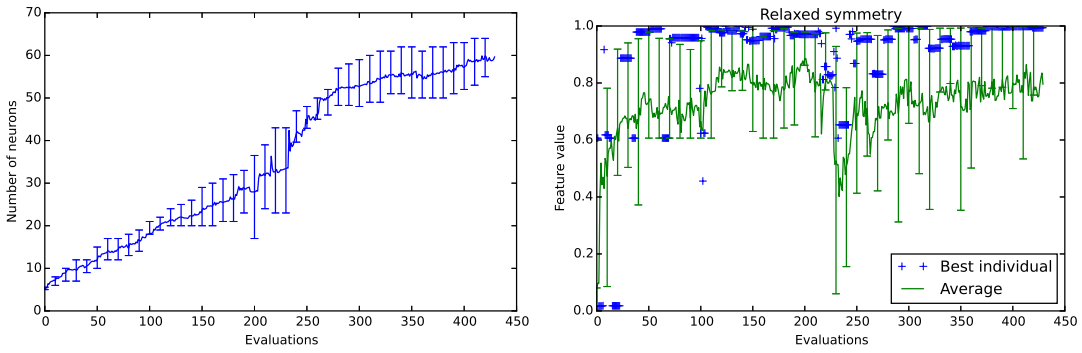
was not more than 0.6. When many images are at the border of the classification, this ratio is very misleading. Curves of all four graphs have a similar shape. From now on, we will only show the classification difference which includes the most information.

Between the 200$^{\text{th}}$ and the 250$^{\text{th}}$ evaluation, we can see a huge drop of the complexity according to the complexity classifier in the figure 4.9c. It was caused by the diversification phase, or precisely, by the *diversify* method. The population was diversified four times and the objective function was simplified by the *simplify* method three times, see the figure 4.8. Afterwards, the inspiration criterion was met, the objective function was modified, and the optimization continued. The graph 4.9e shows a reduction of the total number of windows in the objective function in this period. Nevertheless, the complexity difference was still increasing after this temporal drop.

The graph 4.9e shows the number of windows in the objective function over evaluations. We can deduce from it a period of the diversification phase that took more than one evaluation. Our operators to modify the objective function can remove at most one window and they always add one window. If the window is not removed, the number of windows will increase, otherwise it will not change. Therefore, the number of window cannot get lower after the *inspire* method. It can only decrease by *simplify* method during the diversification phase.

The graph 4.9f shows the description metric values over evaluations. Values are only shown for the evaluations, in which ITS tested the inspiration criterion. If the value is bigger than the threshold I, the inspiration criterion was met and the objective function was modified. The extension and the coverage factor values are shown in 4.9g and 4.9h. If the coverage value is smaller than the removal threshold R, one window may have been removed from the objective function.

The graph 4.10a shows a number of neurons in neural networks of individuals. We can see that NEAT continuously complexifies neural networks. The graph 4.10b shows relaxed symmetry, that was used as feature function by ITS, over evaluations.



(a) Number of neurons in individuals

(b) Feature function used by ITS

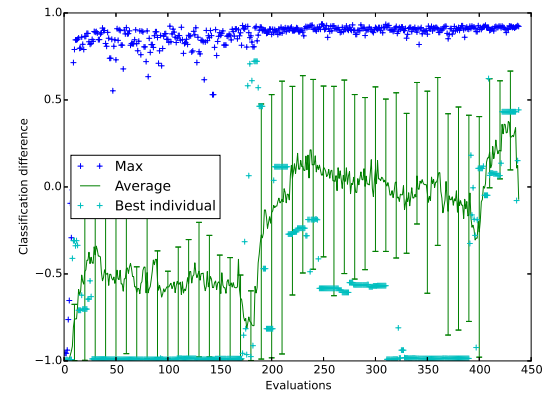
Figure 4.10: Additional information about the selected run of ITS over evaluations. The bottom and the top of error bars are the 25$^{\text{th}}$ and the 75$^{\text{th}}$ percentiles.

4.3 Selected runs

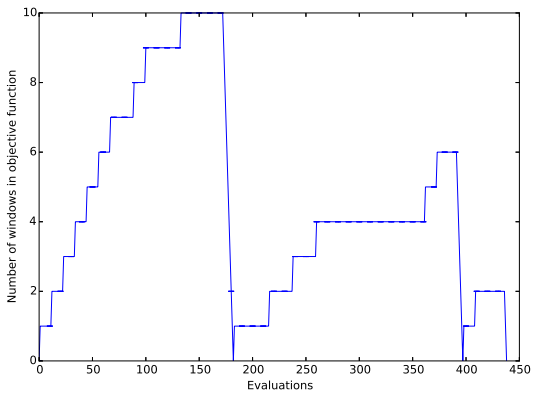
In this section, we describe different behaviours of several selected runs. We use the knowledge built in the previous section, namely, the visual representation of operations in ITS, graphs and their meaning. To recall, I denotes the inspiration threshold in the inspiration criterion, and R denotes the removal threshold to remove a window from the objective function.

4.3.1 Increase of complexity after diversification phase

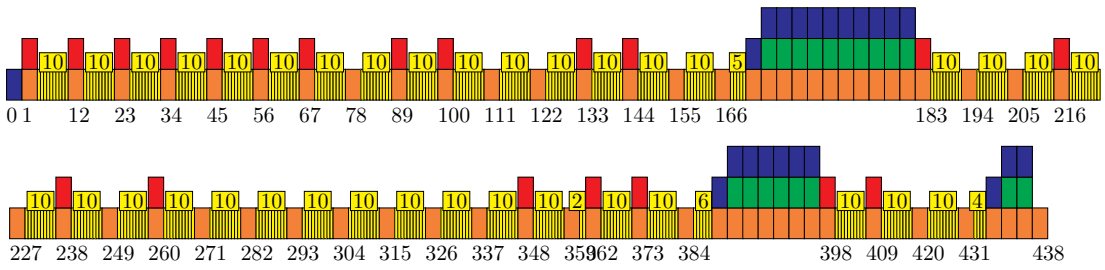
The figure 4.11 shows an example where the diversification phase increased the complexity after a temporal drop according to the complexity classifier. ITS used global contrast factor as the feature function with the penalty, $I = 0.1$, and $R = 0.4$. The temporal drop is the result of the *diversify* method. The figure 4.11c shows that two long diversification phases started at the 172$^{\text{nd}}$ and at the 391$^{\text{st}}$ evaluation. They also removed all windows from the objective function, see the graph 4.11b. Nevertheless, the classification difference values were much higher after the diversification phase in comparison with before, see the graph 4.11a.



(a) Classification difference values



(b) Number of windows in objective function

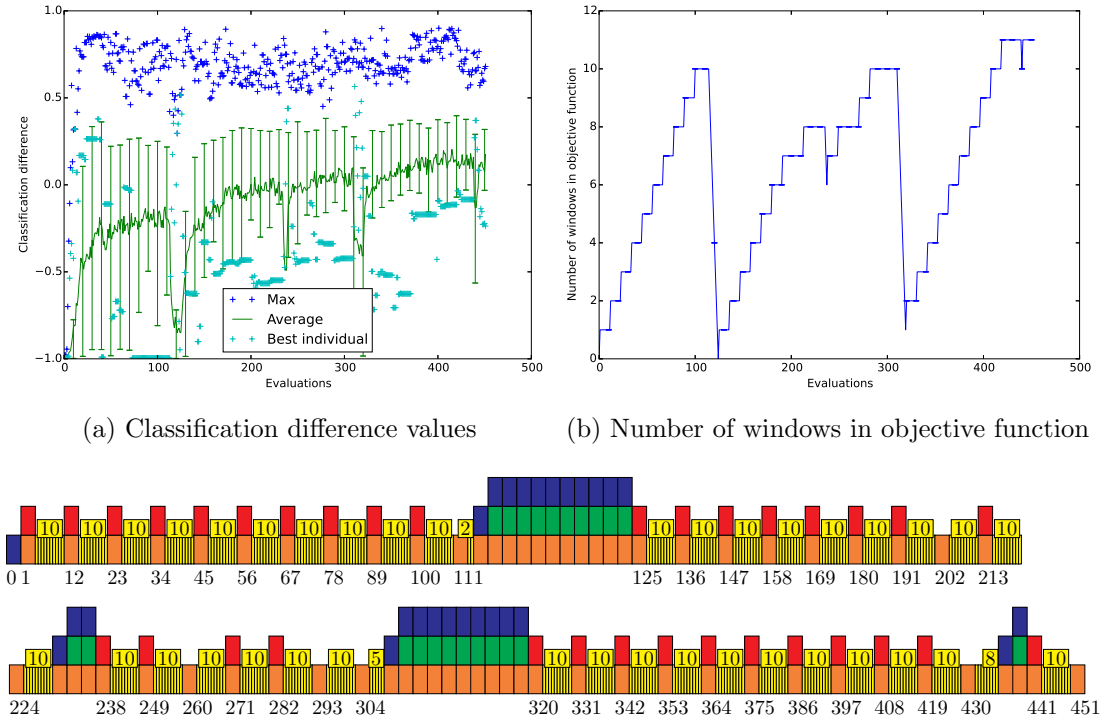


(c) Visualization of operations over evaluations

Figure 4.11: Information about the run of ITS where the diversification phase increased the complexity after a temporal drop according to the complexity classifier. The bottom and the top of error bars are the 25$^{\text{th}}$ and the 75$^{\text{th}}$ percentiles. The figure 4.7 describes the representation of the visualization of operations

4.3.2 No change of complexity after diversification phase

The figure 4.12 shows an example where the diversification phase did not change the complexity too much after a temporal drop according to the complexity classifier. ITS used global contrast factor as the feature function, $I = 0.1$, and $R = 0$. All diversification phases caused a temporal drop of the complexity difference, see the figures 4.12a and 4.12c. Afterwards, the complexity difference almost returned to the same values. The run did not have the operator to remove a window from the objective function. ITS only added windows to the objective function until the convergence of the underlying optimization, see the figure 4.12b.



(a) Classification difference values

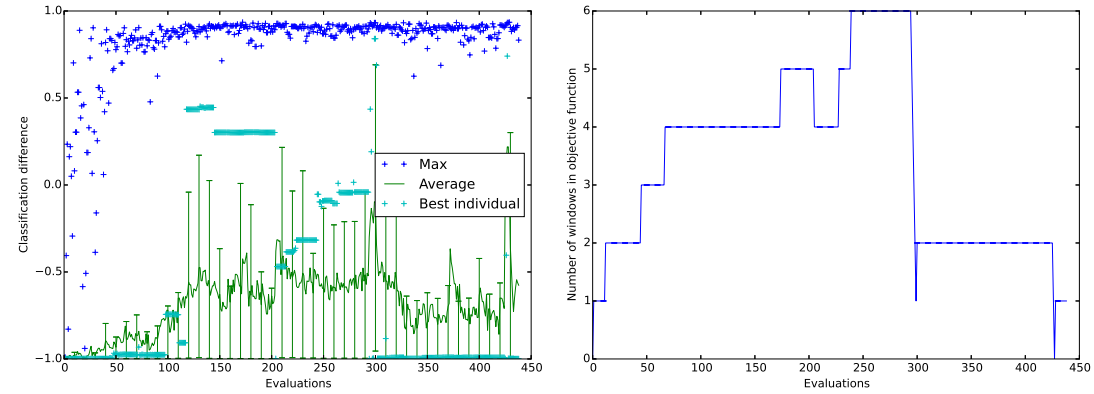
(b) Number of windows in objective function

(c) Visualization of operations over evaluations

Figure 4.12: Information about the run of ITS where the diversification phase almost did not change the complexity after a temporal drop according to the complexity classifier.

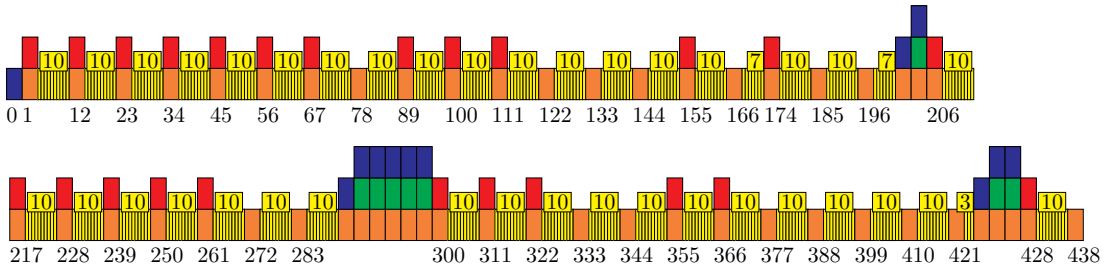
4.3.3 Increase of complexity during diversification phase

The figure 4.13 shows an example where the diversification phase increased the complexity difference while diversifying the population. ITS used normalized variance as the feature function, $I = 0.02$, and $R = 0.3$. The diversification phase started in the 294^{th} evaluation and the diversification of the population significantly increased the complexity difference, see the rapid increase of the average classification difference in the figure 4.13a. Afterwards, the complexity difference dropped back to similar values as before the diversification phase.



(a) Classification difference values

(b) Number of windows in objective function



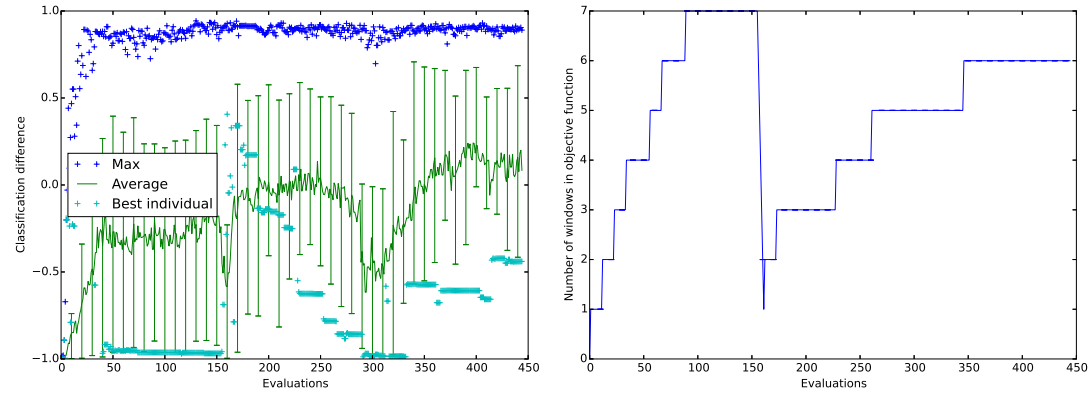
(c) Visualization of operations over evaluations

Figure 4.13: Information about the run of ITS where the diversification phase increased the complexity difference while diversifying the population.

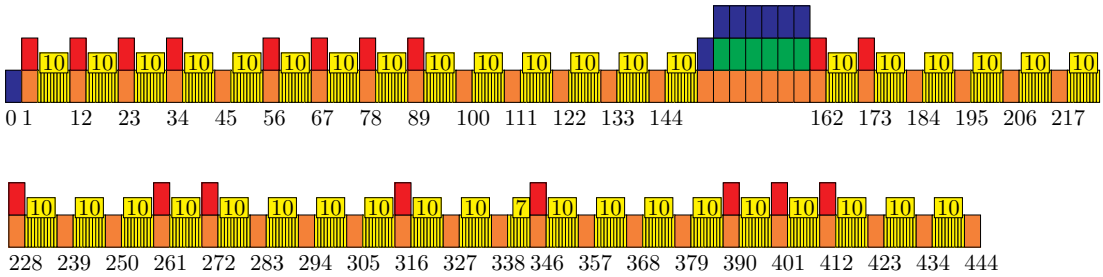
4.3.4 Importance of objective function to complexity

The performance of ITS also depends on the objective function. The change of the objective function can possibly increase or decrease the complexity of images according to the complexity classifier. For instance, the figure 4.14 shows one run where this modification significantly decreased the complexity difference. ITS used global contrast factor as the feature function with the penalty, $I = 0.1$, and $R = 0.4$. The graph 4.14a shows a significant drop around the 300^{th} evaluation, but the figure 4.14c shows no diversification at that period. In the 272^{nd} evaluation, the objective function was modified and the complexity difference started to decrease. After another modification in the 316^{th} evaluation, the complexity difference started to increase again.

Another example shows the figure 4.15 where the change of objective function rapidly changed the complexity of images according to the complexity classifier. The run used normalized variance, relaxed symmetry, and global contrast factor as feature functions with the penalty, $I = 0.1$, and $R = 0$. The change of the objective function in the 55^{th} evaluation rapidly decreased the complexity difference. In the 350^{th} evaluation, the change significantly increased the complexity difference. This run had high value of I, therefore it was difficult to meet the inspiration criterion. The graph 4.15b shows that ITS used at most one window and the diversification phase always removed the window from the objective function. Furthermore, ITS used 3 feature functions and the choice of feature for only one window had high influence on the complexity difference.

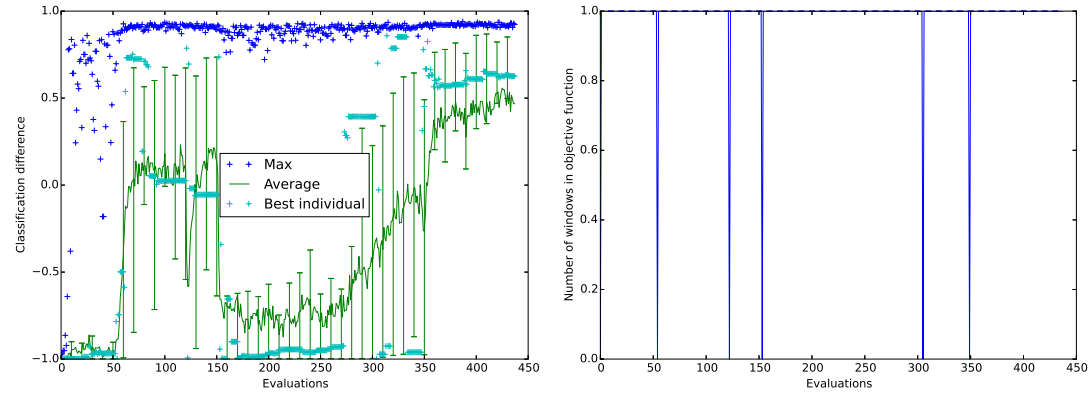


(a) Classification difference values (b) Number of windows in objective function

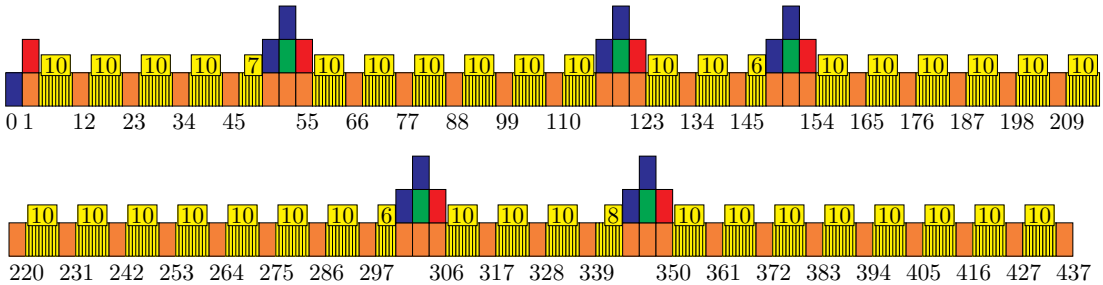


(c) Visualization of operations over evaluations

Figure 4.14: Information about the run of ITS where one modification of the objective function significantly decreased the complexity difference.



(a) Classification difference values (b) Number of windows in objective function

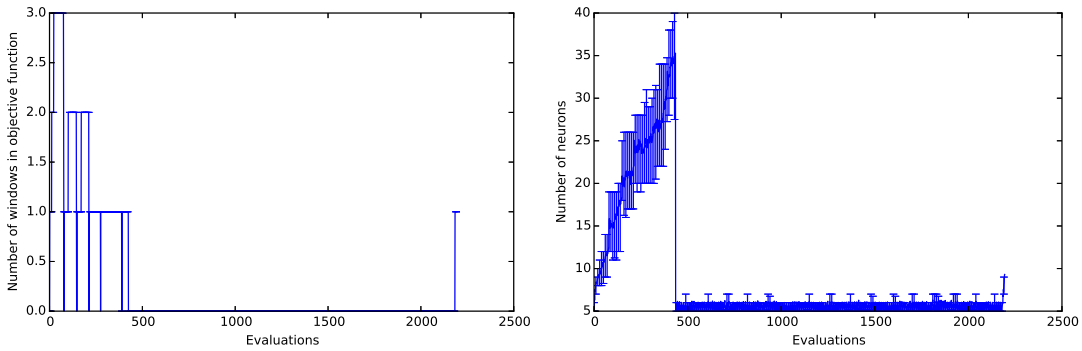


(c) Visualization of operations over evaluations

Figure 4.15: Information about the run of ITS where one modification of the objective function significantly changed the complexity difference.

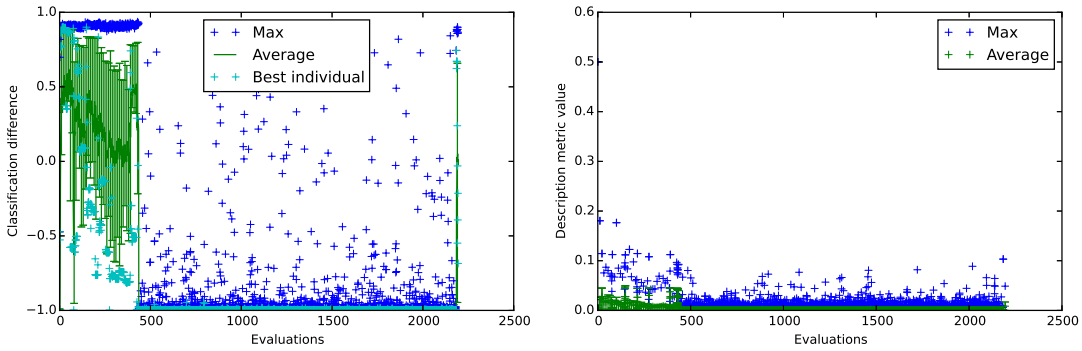
4.3.5 Inability to leave diversification phase

We encountered a few examples in which ITS was not able to leave the diversification phase for a long time. For instance, the figure 4.16 shows the run with the following setting: Tenengrad as the feature function with the penalty, $I = 0.1$, and $R = 0.4$. The graph 4.16a shows that ITS was in the diversification phase from the 415^{th} to the 2184^{th} evaluation. This is very improbable situation, because NEAT is always reset after 10 tries with an empty objective function. The graph 4.16b shows this behaviour. The graph 4.16c shows that the population consisted mainly of simple images the whole time according to the complexity classifier. The inspiration criterion were not met because the description metric values were below the threshold I, see the graph 4.16d.



(a) Number of windows in objective function

(b) Number of neurons in individuals



(c) Classification difference values

(d) Description metric values

Figure 4.16: Information about the run where ITS was not able to leave the diversification phase for a long time.

4.4 Comparison

In this section, we compare ITS with different settings and with the uninspiring search using the same set of feature functions. We can aggregate all trials of ITS using the same setting by evaluations. Nevertheless, it is unfair to ITS, because each run is unique. ITS contains different phases and it is influenced by the current population and the current objective function. In the particular evaluation, one trial may optimize by the objective function, but another trial may diversify the population, which may result in decreasing a "complexity"

of images. They also differ by their objective functions and even by the total number of windows in them. However, if we are able to show that ITS creates "more complex" images in comparison with the uninspired search even with this handicap, we will accomplish our goal. We can compare them by two presented methods: the complexity test by optimization and the complexity classifier. Both of these methods have many disadvantages. Nevertheless, they are better than only our subjective visual comparison of produced images.

For the complexity test by optimization, we selected representative images from both algorithms. Images from the uninspired search were chosen from the best individuals of last generations. Images from ITS could be possibly chosen from any evaluation. However, one trial of ITS with 40 creative cycles is on average about 400 evaluations and each population contains at least 150 individuals. Therefore, the result of one trial is 60000 images. We selected images from the evaluations that could be considered as important. That are the evaluations when the objective function was modified and when the optimization converged. Additionally, we did not select too visually complex images, because we wanted to give a chance to the direct optimization. This choice was based on our preliminary experiments. Nevertheless, we will show that it probably was a mistake. The selected images were used for the complexity test by optimization. The pixel distances of the best individuals from last generations of all trials were aggregated. We can statistically compare the results of two algorithms by Wilcoxon rank sum test. We will always use this test with the one sided alternative hypothesis and specify the number of trials used in each data set. The algorithm with higher values can be considered as producing more complex images. However, we showed limitations of this test in the figure 4.1. To recall, when the pixel distance is lower than 16000, the image is classified as simple, otherwise it is labelled as complex image. Therefore, we also classify the algorithm by the produced images. The problematic situation is when both algorithms produce images classified as complex.

Each run of ITS can have a different total number of evaluations. We choose the minimal number from the last evaluations of all trials as the evaluation for the comparison. For the comparison by the complexity classifier, the complexity difference values of all individuals from the previously selected evaluation of all trials of ITS were aggregated. For uninspired search, the complexity difference values of all individuals from the last generation of all trials were aggregated. We can again statistically compare the results of two algorithms by Wilcoxon rank sum test with the one sided alternative hypothesis. The algorithm with higher values can be consider as producing more images classified as complex in the population. The idea is that the algorithm with higher values has a higher probability to generate an image classified as complex.

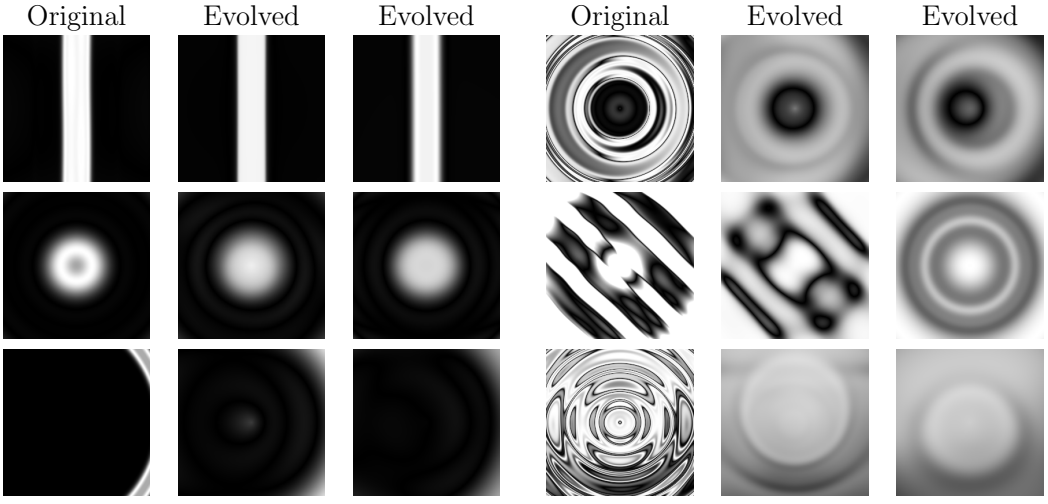
We tested the uninspired search using a set of features applied to the whole image for 700 generations in our experiments. We found this number of generations sufficient from our preliminary experiments, in which we tested the optimization until the stagnation for 50 generations in 10 trials. The number of last generations were as follows: global contrast factor $\sim 453.9 \pm 221.5$, relaxed symmetry $\sim 87.2 \pm 14.7$, normalized variance $\sim 706.0 \pm 379.6$, Tenengrad $\sim 810.7 \pm 284.7$, choppiness $\sim 721.6 \pm 335.2$, and image complexity by JPEG compression $\sim 318.6 \pm 96.0$. The generations after 700 improved the best fitness

value only by a little and the resulting images did not almost visually differ.

We used computers with Intel® Xeon® Processor E5506 (2.13 GHz) for our experiments. Each computation used only one core, thus we are able to run many trials in parallel on a cluster. Nevertheless, the bigger neural network the more computationally it is to create the image and NEAT complexifies neural networks over generations. Due to this problem, we were not able to run statistically enough trials for everything. For instance, we only tested one image in the complexity test by optimization in 5 trials, which is not statistically enough. On the other hand, we gained more samples by combining the results of more images from the tested algorithm.

4.4.1 Normalized Variance

We tested uninspired search for 12 trials (one trial $\sim 75.5 \pm 4.9$ hours). The resulting images were visually very simple, see the figure 3.5 for examples. We selected 7 images from best individuals of last generations and used them for the complexity test by optimization in 2 trials. Additionally, we selected 2 images based on the results, that were the most difficult to evolve, to get more statistically significant results and tested them again for 5 trials. In the end, we used the complexity test by optimization in 24 trials (one trial $\sim 42.0 \pm 13.6$ hours) with 2 images, each for 7 trials, and 5 images, each for 2 trials. The images were able to be found by the optimization, therefore images produced by uninspired search are classified as simple according to the complexity test by optimization, see the figure 4.18. The figure 4.17a shows examples of evolved images.



(a) Uninspired search (b) ITS

Figure 4.17: Examples of images used for the complexity test by optimization. Shown evolved images are visually the most similar to the original image.

We tested ITS using the inspiration threshold $I \in \{0.01, 0.02\}$, and the removal threshold $R \in \{0, 0.2, 0.3\}$. The figure 4.20a shows the influence of the particular setting on the total number of windows in the objective function. We tested each settings for 10 trials (1 trial $\sim 16.5 \pm 9.6$ hours). The figure 4.21 shows the diversity of evolved images.

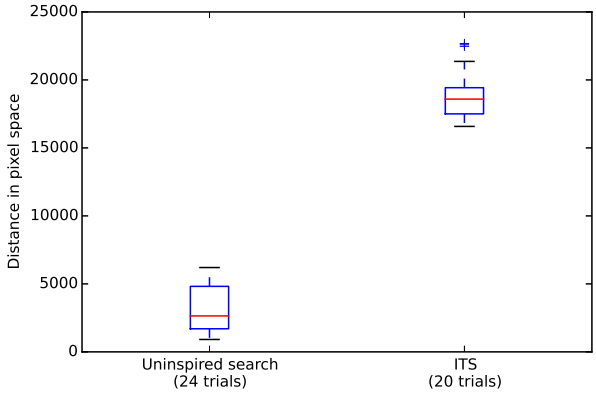


Figure 4.18: Comparison of ITS with uninspired search by the complexity test by optimization using normalized variance as feature function.

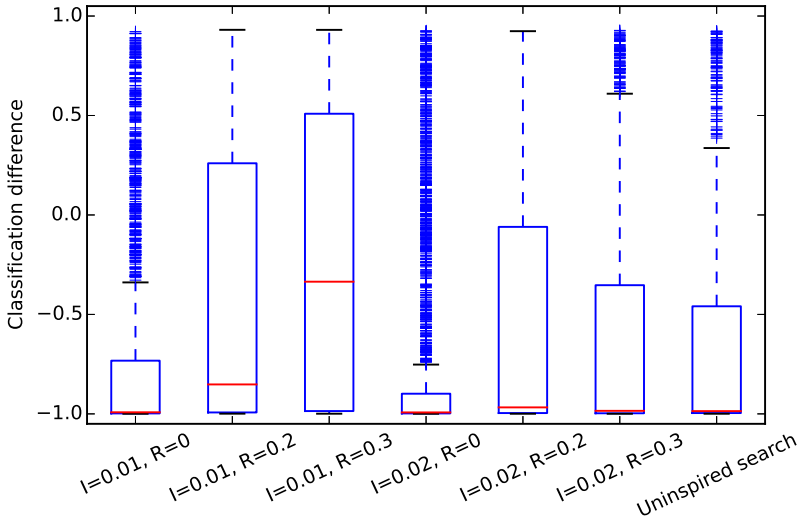
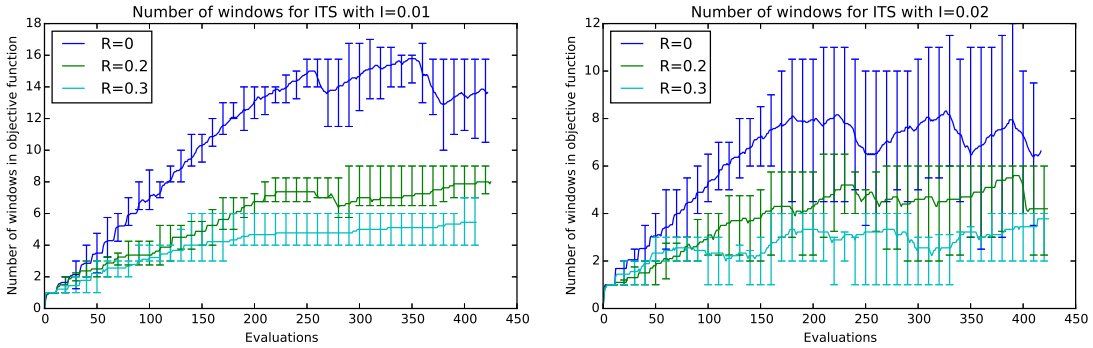


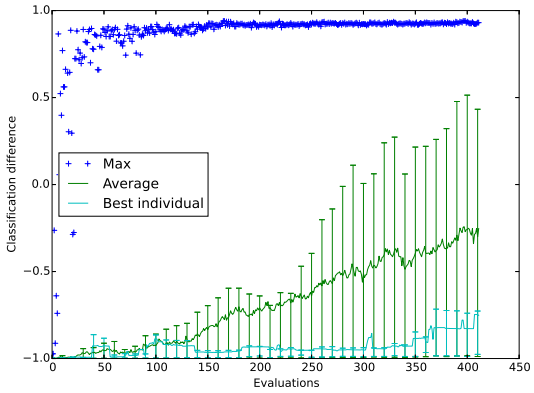
Figure 4.19: Comparison of ITS with different settings and with uninspired search by the complexity classifier using normalized variance as feature function.

We selected 4 images from all trials and used them for the complexity test by optimization in 5 trials. The result is 20 trials (one trial $\sim 58.0 \pm 17.7$ hours). The images were not able to be found by the optimization, therefore images produced by ITS are classified as complex according to the complexity test by optimization, see the figure 4.18. The figure 4.17b shows examples of evolved images. We can compare ITS with uninspired search by the best individuals from last generations, see the figure 4.18. ITS produced significantly more complex images ($p < 0.001$) according to the complexity test by optimization.

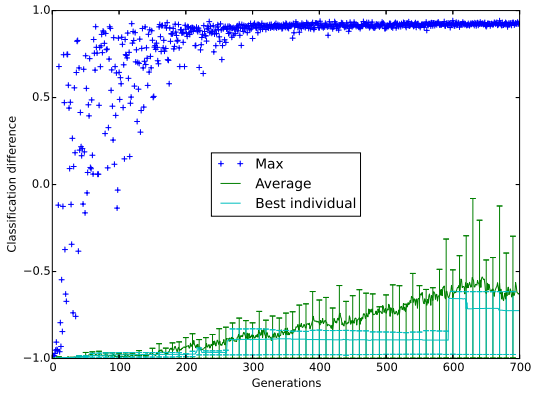
We can compare different settings of ITS by the complexity classifier, see the figure 4.19. The removal of a window with each settings produced significantly more complex images ($p < 0.001$ for each combination). We selected the setting using $I = 0.01$, $R = 0.3$. ITS produced significantly more complex images ($p < 0.001$) according to the complexity classifier. Figures 4.20b, 4.20c compare the classification difference and figures 4.20d, 4.20e compare normalized variance values of ITS with uninspired search over time.



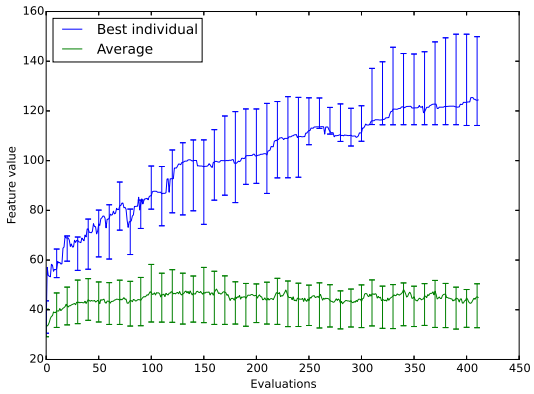
(a) Number of windows in the objective function for different settings in ITS.



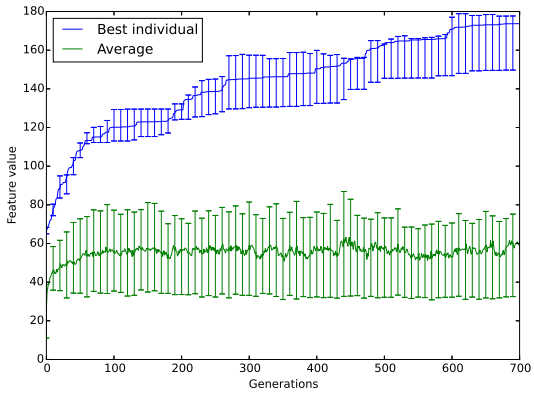
(b) Classification difference in ITS



(c) Classification difference in uninspired search



(d) Normalized varince in ITS



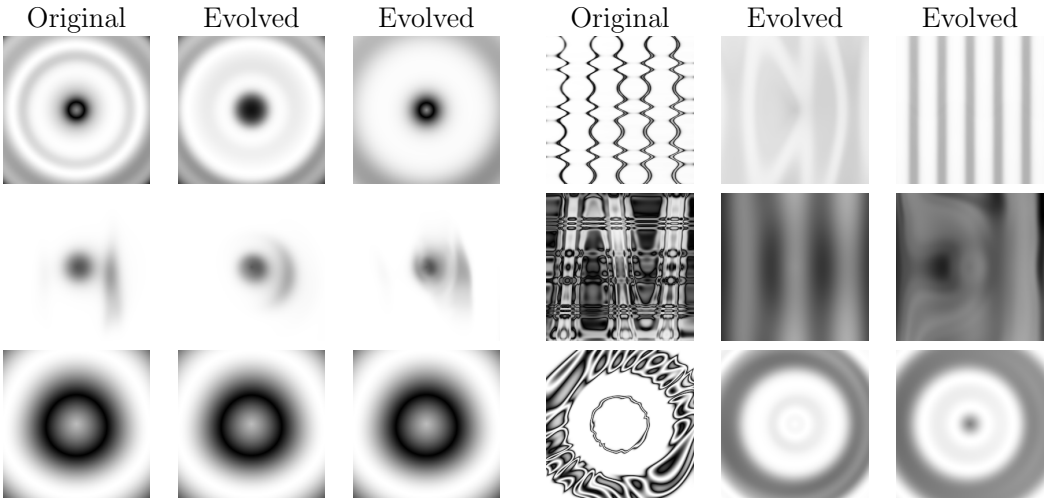
(e) Normalized varince in uninspired search

Figure 4.20: Comparison of ITS with different settings and with uninspired search using normalized variance as feature function. Except the figure (a), ITS used the selected setting of $I = 0.01$, $R = 0.3$.

Figure 4.21: Examples of images found by ITS using normalized variance as feature function with their complexity classifications according to the complexity classifier. The classification consists of the simplicity and the complexity node.

4.4.2 Relaxed symmetry

We tested uninspired search for 15 trials (one trial $\sim 188.9 \pm 32.2$ hours). The generated images were visually very simple, see the figure 3.4 for examples. We selected 10 images from best individuals of last generations and used them for the complexity test by optimization in 2 trials. Additionally, we selected 2 images based on the results, that were the most difficult to evolve, to get more statistically significant results and tested them again for 5 trials. In the end, we used the complexity test by optimization in 30 trials (one trial $\sim 36.6 \pm 10.8$ hours) with 2 images, each for 7 trials, and 8 images, each for 2 trials. The images were able to be found by the optimization, therefore images produced by uninspired search are classified as simple according to the complexity test by optimization, see the figure 4.23. The figure 4.22a shows examples of evolved images.



(a) Uninspired search (b) ITS

Figure 4.22: Examples of images used for the complexity test by optimization. Shown evolved images are visually the most similar to the original image.

We tested ITS using the inspiration threshold $I \in \{0.01, 0.02\}$, and the removal threshold $R \in \{0, 0.3, 0.4\}$. The figure 4.25a shows the influence of the particular setting on the total number of windows in the objective function. We tested each settings for 10 trials (1 trial $\sim 17.0 \pm 8.8$ hours). The figure 4.26 shows the diversity of evolved images.

We selected 3 images from all trials and used them for the complexity test by optimization in 5 trials. The result is 15 trials (one trial $\sim 55.0 \pm 12.9$ hours). The images were not able to be found by the optimization, therefore images produced by ITS are classified as complex according to the complexity test by optimization, see the figure 4.23. The figure 4.22b shows examples of evolved images. We can compare ITS with uninspired search by the best individuals from last generations, see the figure 4.23. ITS produced significantly more complex images ($p < 0.001$) according to the complexity test by optimization.

We can compare the different settings by the complexity classifier, see the figure 4.24. The removal of a window with each settings produced significantly more complex images ($p < 0.001$ for each combination). We selected the setting using $I = 0.01$, $R = 0.4$ ITS produced significantly more complex images ($p <$

0.001) according to the complexity classifier. Figures 4.25b, 4.25c compare the classification difference and figures 4.25d, 4.25e compare relaxed symmetry values of ITS with uninspired search over time.

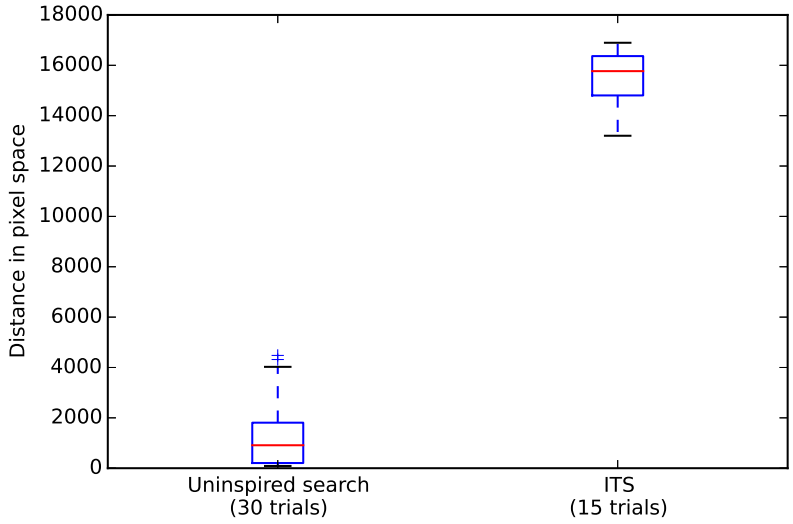


Figure 4.23: Comparison of ITS with uninspired search by the complexity test by optimization using relaxed symmetry as feature function.

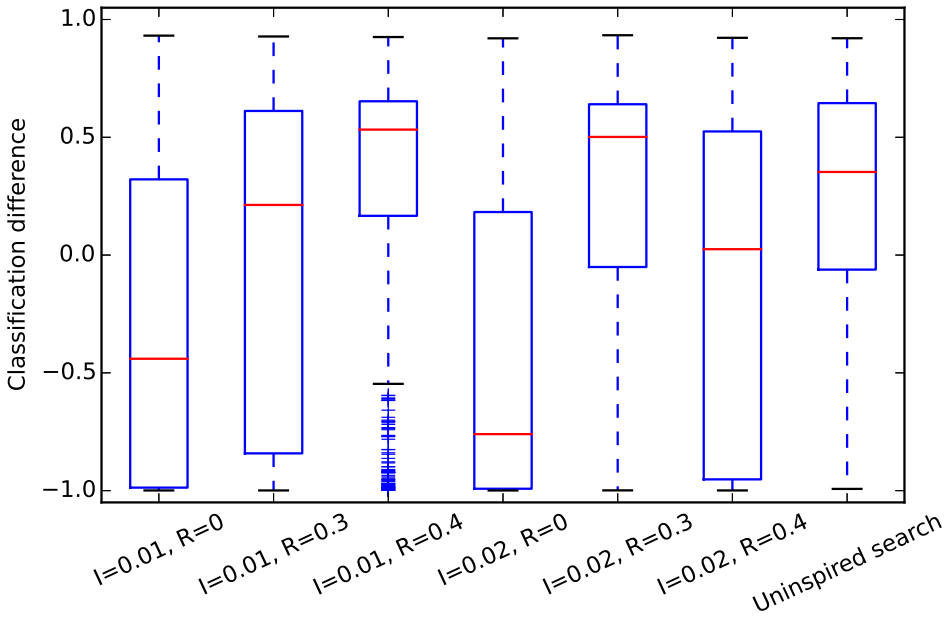
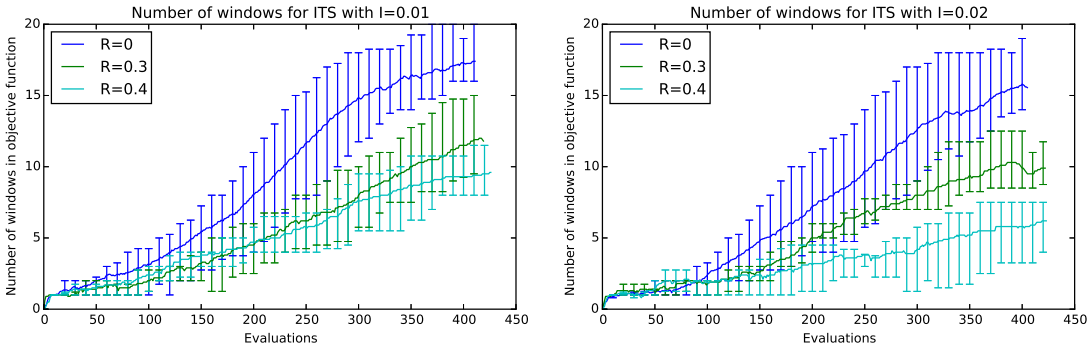
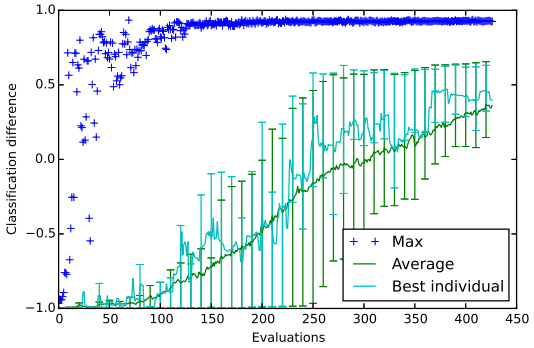


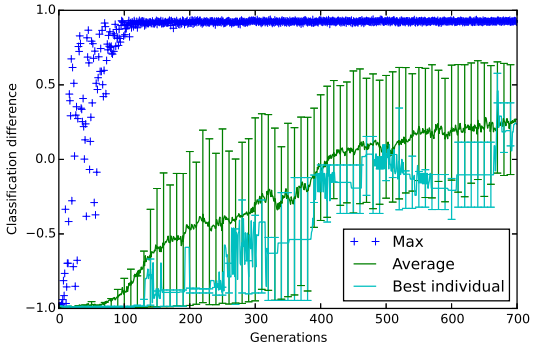
Figure 4.24: Comparison of ITS with different settings and with uninspired search by the complexity classifier using relaxed symmetry as feature function.



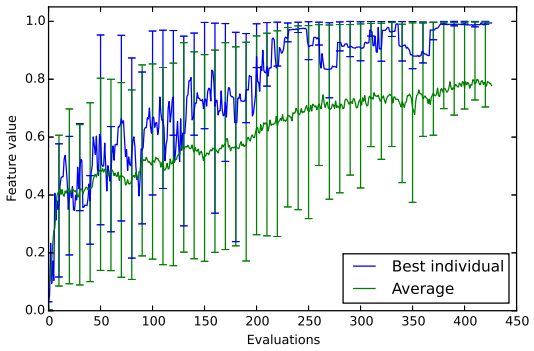
(a) Number of windows in the objective function for different settings in ITS.



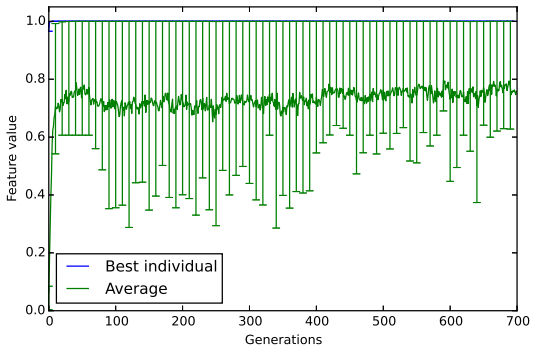
(b) Classification difference in ITS



(c) Classification difference in uninspired search



(d) Relaxed symmetry in ITS



(e) Relaxed symmetry in uninspired search

Figure 4.25: Comparison of ITS with different settings and with uninspired search using relaxed symmetry as feature function. Except the figure (a), ITS used the selected setting of $I = 0.01$, $R = 0.4$.

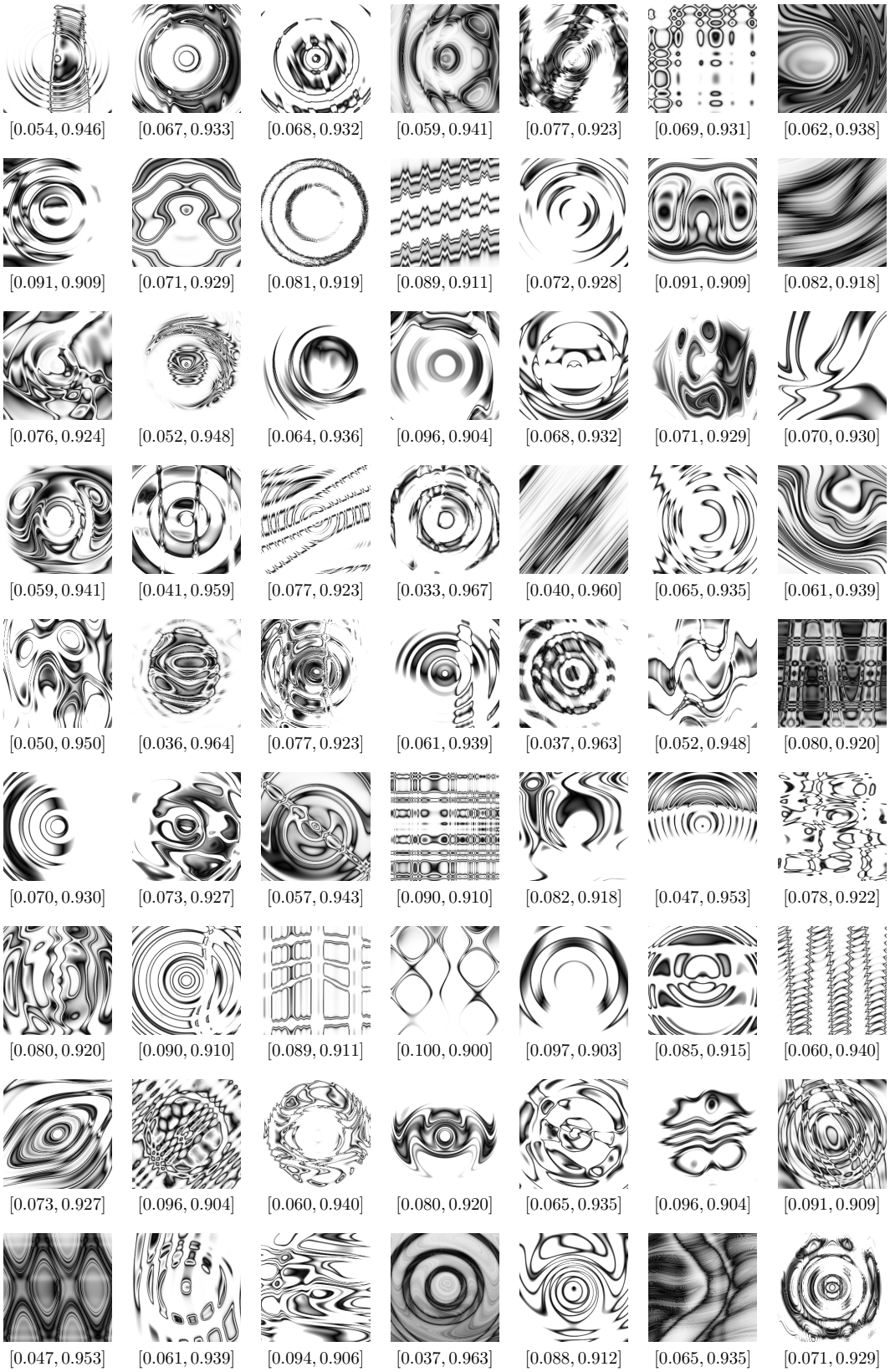


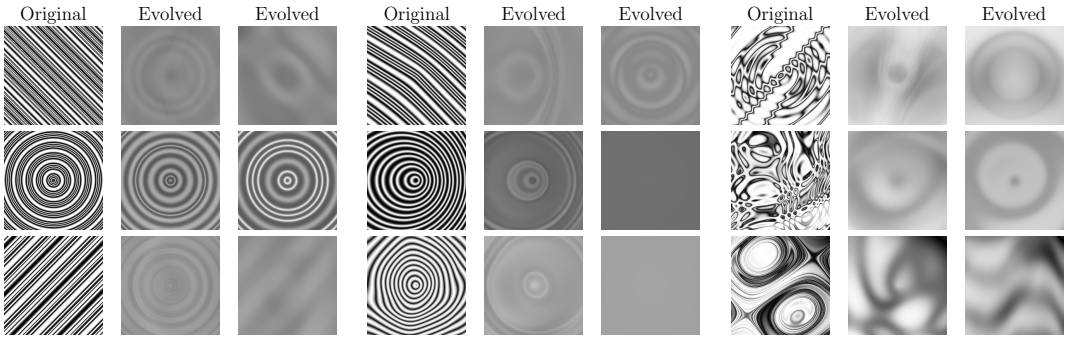
Figure 4.26: Examples of images found by ITS using relaxed symmetry as feature function with their complexity classifications according to the complexity classifier. The classification consists of the simplicity and the complexity node.

4.4.3 Global contrast factor

We tested uninspired search for 10 trials. The generated images were visually very simple, see the figure 3.3 for examples. We selected 10 images from best individuals of last generations and used them for the complexity test by optimization in 2 trials. Additionally, we selected 1 image based on the results, that was the most difficult to evolve, to get more statistically significant results and tested it again for 5 trials. The result is 25 trial (one trial $\sim 57.8 \pm 15.7$ hours) with 1 image for 7 trials, and 9 images, each for 2 trials. Due to the problem of high spacial frequencies, we also tested uninspired search with the penalty for 5 trials (one trial $\sim 63.6 \pm 11.5$ hours). The generated images were again visually simple, see the figure 4.27 for examples. We selected 5 images for the complexity test. The result is 15 trials (one trial $\sim 64.0 \pm 15.5$ hours) with 5 images, each for 3 trials. The images from both settings were not able to be found by the optimization, therefore images produced by uninspired search are classified as complex according to the complexity test by optimization, see the figure 4.29. The figures 4.28a and 4.28b show examples of evolved images.

[0.937, 0.063] [0.994, 0.006] [0.994, 0.006] [0.983, 0.017] [0.997, 0.003]

Figure 4.27: Examples of images found by uninspired search using global contrast factor as feature function with the penalty with their complexity classifications according to the complexity classifier. Examples of images found without the penalty are shown in the figure 3.3.



(a) Uninspired search (b) Uninspired search with the penalty (c) ITS

Figure 4.28: Examples of images used for the complexity test by optimization. Shown evolved images are visually the most similar to the original image.

We tested ITS using the inspiration threshold $I = 0.1$, the removal threshold $R \in \{0, 0.4, 0.5\}$, and with and without the penalty. The figure 4.30a shows the influence of the particular setting on the total number of windows in the objective function. We tested each settings for 10 trials (1 trial $\sim 15.7 \pm 4.3$ hours). The figure 4.31 shows the diversity of evolved images.

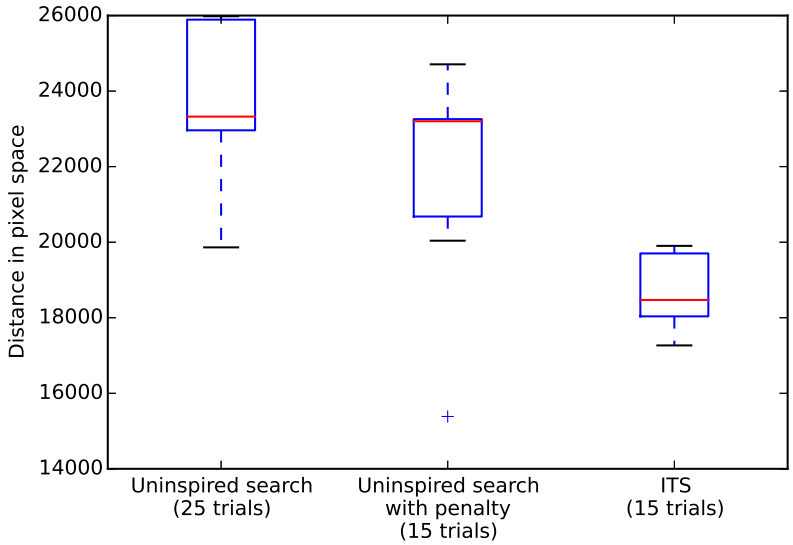
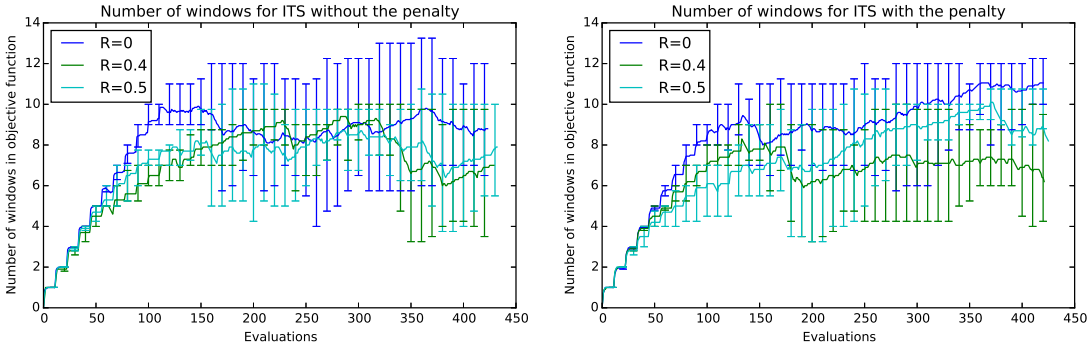


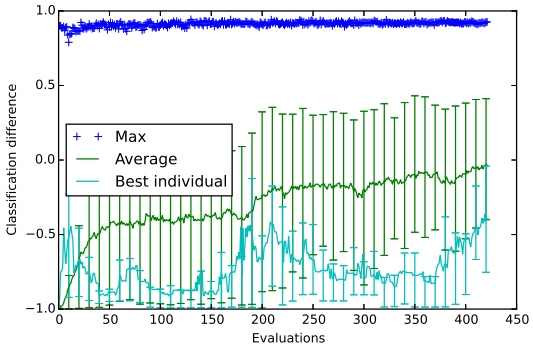
Figure 4.29: Comparison of ITS with uninspired search by the complexity test by optimization using global contrast factor as feature function.

We selected 3 images from all trials and used them for the complexity test by optimization in 5 trials. The result is 15 trials (one trial $\sim 56.8 \pm 18.7$ hours). The images were not able to be found by the optimization, therefore images produced by ITS are classified as complex according to the complexity test by optimization, see the figure 4.29. The figure 4.28c shows examples of evolved images. We can compare ITS with uninspired search by the best individuals from last generations, see the figure 4.29. Uninspired search with and without the penalty produced more complex images ($p < 0.001$ for both) according to the complexity test by optimization. However, both of them produced complex images according to the complexity test. This is the case, where the complexity test does not work. The images produced by uninspired search are subjectively more simple and we think they are from the class II or IV (see the figure 4.2).

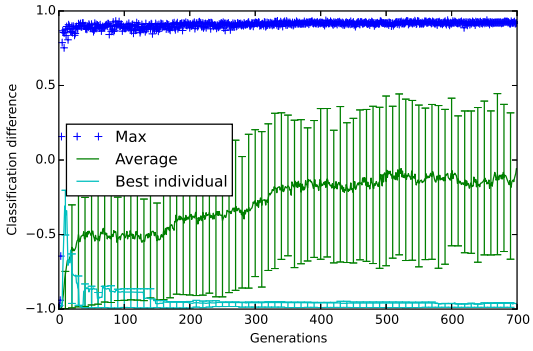
We selected the setting using $I = 0.1$, $R = 0.4$ and with the penalty in order to compare it with uninspired search, which also used the penalty. The result is not conclusive ($p = 0.164$). We cannot reject hypothesis that both of them produced images of the same complexity according to the complexity classifier. Nevertheless, we find images from ITS visually more interesting, compare the figure 4.31 with the figure 4.27. Figures 4.30b, 4.30c compare the classification difference, figures 4.30d, 4.30e compare global contrast factor values, and figures 4.30f, 4.30g compare penalty values of ITS with uninspired search over time.



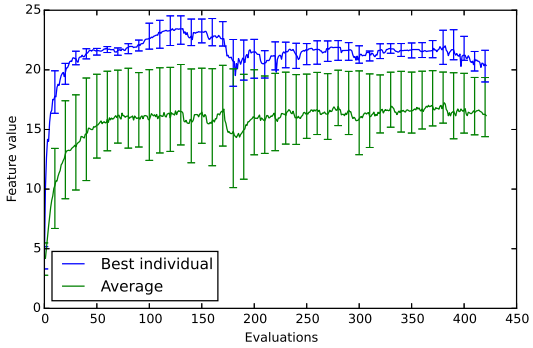
(a) Number of windows in the objective function for different settings in ITS.



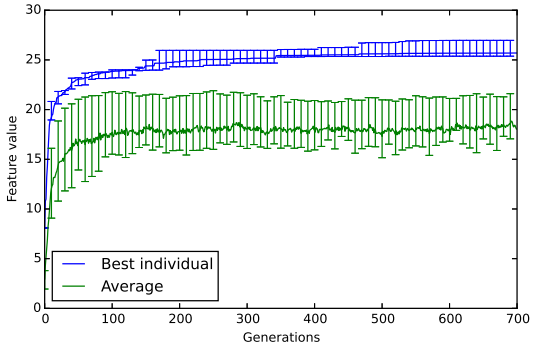
(b) Classification difference in ITS



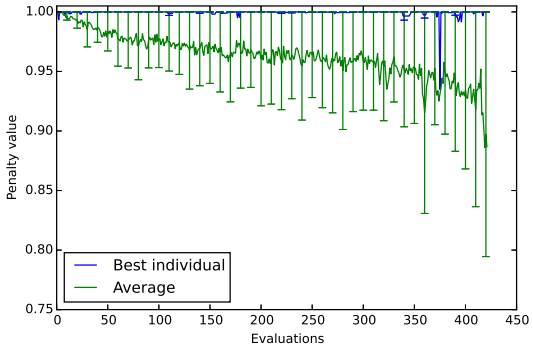
(c) Classification difference in uninspired search



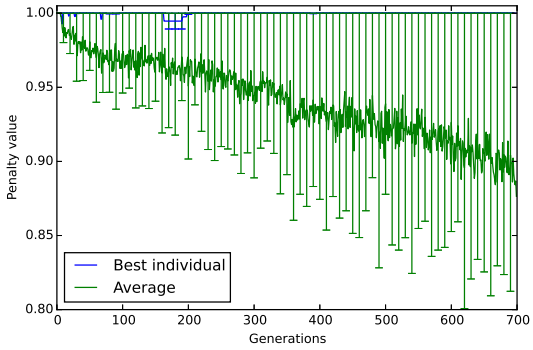
(d) Global contrast factor in ITS



(e) Global contrast factor in uninspired search



(f) Penalty in ITS



(g) Penalty in uninspired search

Figure 4.30: Comparison of ITS using $I = 0.1$, $R = 0.4$ with uninspired search using global contrast factor as feature function with the penalty.

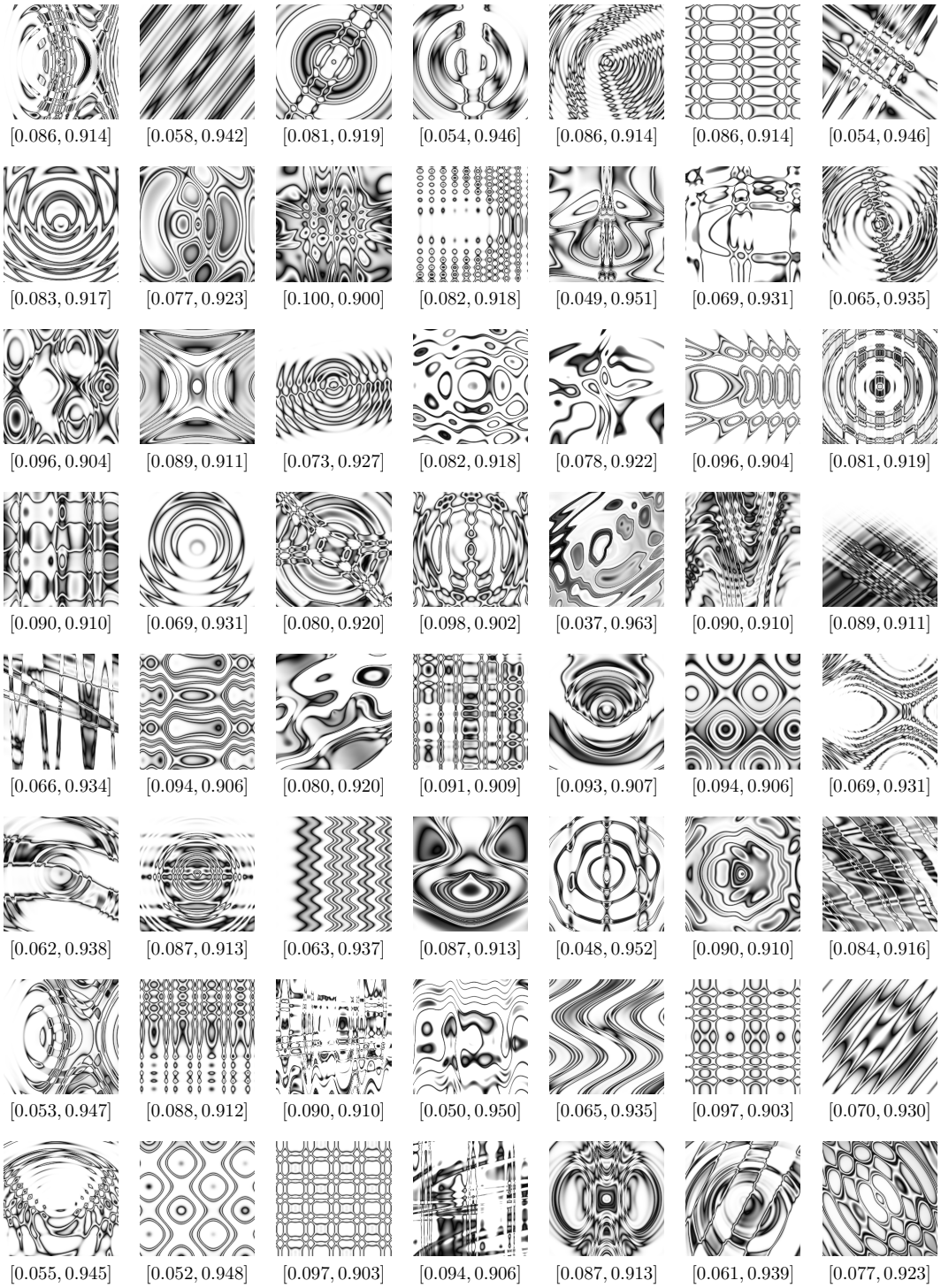


Figure 4.31: Examples of images found by ITS with global contrast factor as feature function with their complexity classifications according to the complexity classifier. The classification consists of the simplicity and the complexity node.

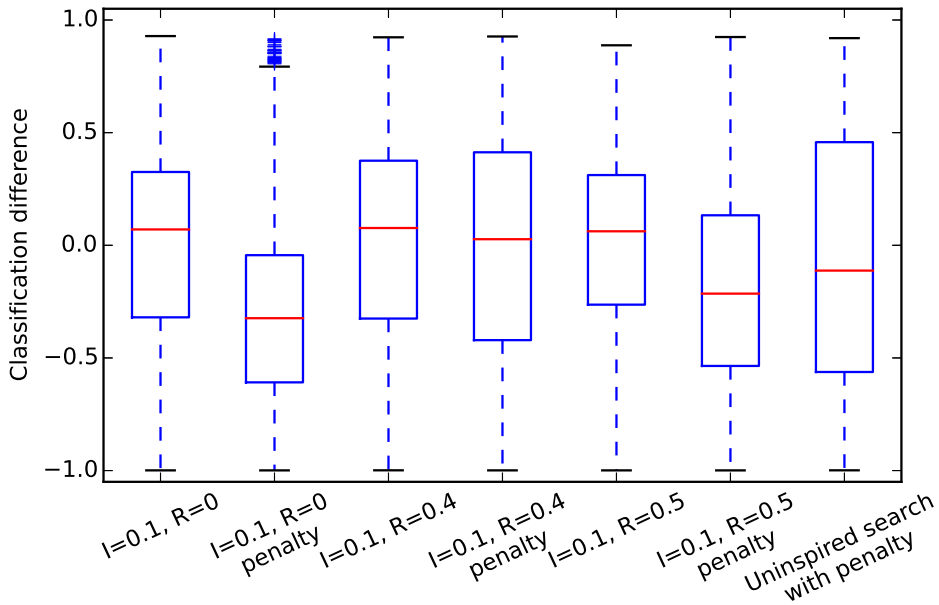


Figure 4.32: Comparison of ITS with different settings and with uninspired search by the complexity classifier using global contrast factor as feature function.

4.4.4 Tenengrad

We tested uninspired search with the penalty for 5 trials (one trial $\sim 65.1 \pm 17.3$ hours). The generated images were visually simple, see the figure 4.33 for examples with the penalty and compare with the figure 3.6 for examples without the penalty. We selected 5 images for the complexity test by optimization. The result is 15 trials (one trial $\sim 58.3 \pm 17.8$ hours) with 5 images, each for 3 trials. The images were not able to be found by the optimization, therefore images produced by uninspired search are classified as complex according to the complexity test by optimization, see the figure 4.35. The figure 4.34a shows examples of evolved images.

We tested ITS using the inspiration threshold $I = 0.04$, the removal threshold $R \in \{0, 0.4\}$, and with and without the penalty. The figure 4.36a shows the influence of the particular setting on the total number of windows in the objective function. We tested each settings for 10 trials (1 trial $\sim 18.4 \pm 14.4$ hours). The

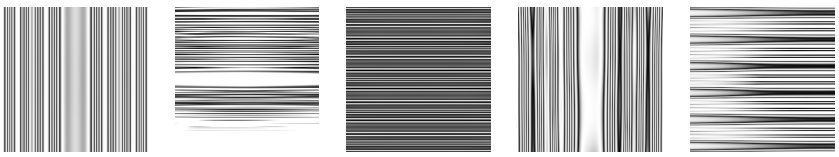
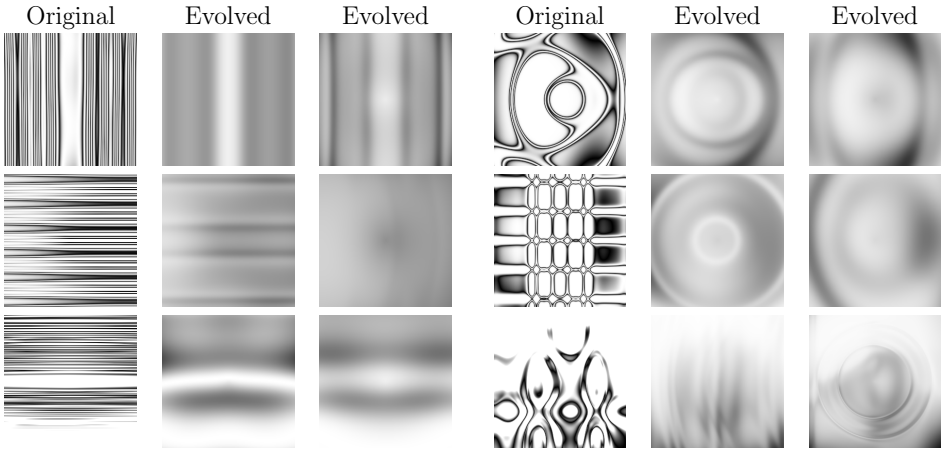


Figure 4.33: Examples of images found by uninspired search using Tenengrad as feature function with the penalty with their complexity classifications according to the complexity classifier. Examples of images found without the penalty are shown in the figure 3.6.



(a) Uninspired search with the penalty (b) ITS

Figure 4.34: Examples of images used for the complexity test by optimization. Shown evolved images are visually the most similar to the original image.

figure 4.37 shows the diversity of evolved images.

We selected 3 images from all trials and used them for the complexity test by optimization in 5 trials. The result is 15 trials (one trial $\sim 50.5 \pm 16.8$ hours). The images were not able to be found by the optimization, therefore images produced by ITS are classified as complex according to the complexity test by optimization, see the figure 4.35. The figure 4.34b shows examples of evolved images. We can compare ITS with uninspired search by the best individuals from last generations, see the figure 4.35. Uninspired search with the penalty produced more complex images ($p < 0.001$) according to the complexity test by optimization. This is the same situation as using global contrast factor as feature function.

We selected the setting using $I = 0.04$, $R = 0.4$ and with the penalty in order to compare it with uninspired search which also used the penalty. The result is not conclusive ($p = 0.4916$). We cannot reject hypothesis that both of them produced images of the same complexity according to the complexity classifier. Nevertheless, we find images from ITS visually more interesting, compare the figure 4.37 with the figure 4.33. Figures 4.36b, 4.36c compare the classification difference, figures 4.36d, 4.36e compare Tenengrad values, and figures 4.36f, 4.36g compare penalty values of ITS with uninspired search over time.

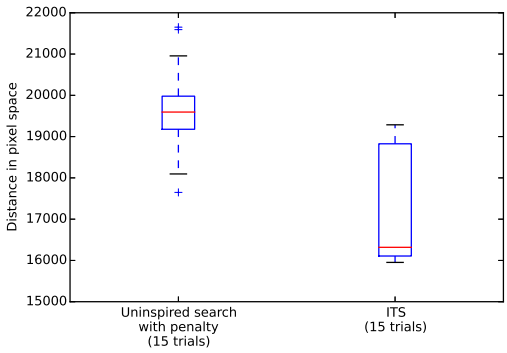
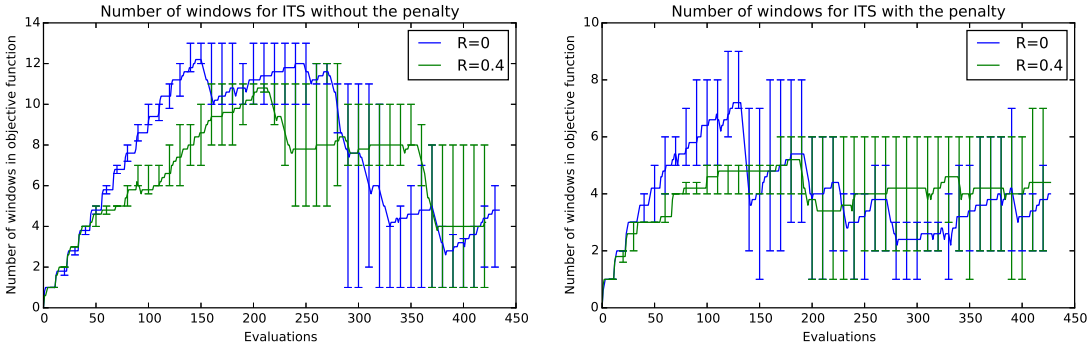
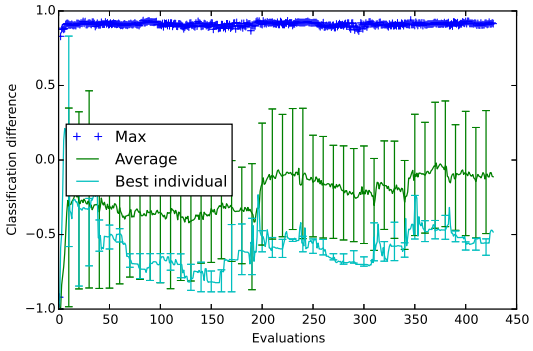


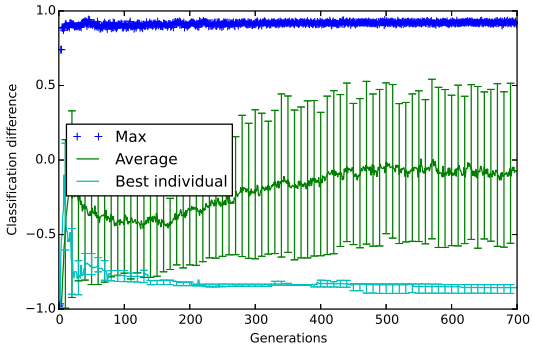
Figure 4.35: Comparison of ITS with uninspired search by the complexity test by optimization using Tenengrad as feature function.



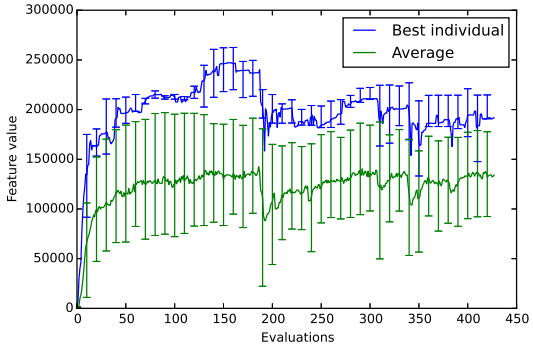
(a) Number of windows in the objective function for different settings in ITS.



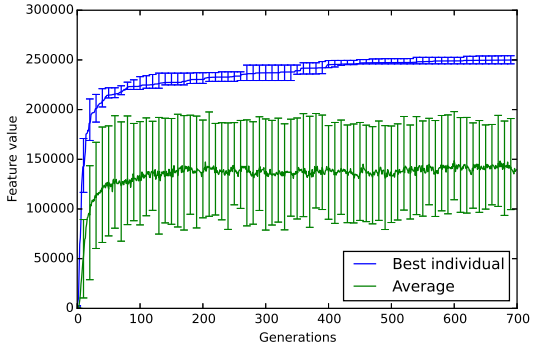
(b) Classification difference in ITS



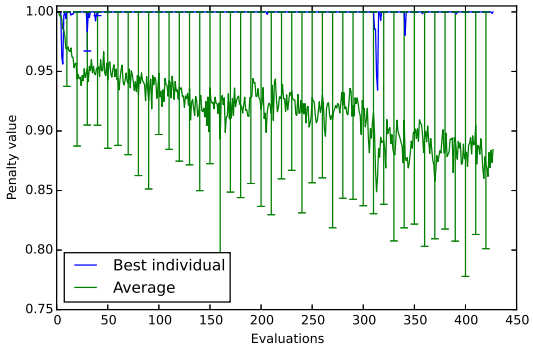
(c) Classification difference in uninspired search



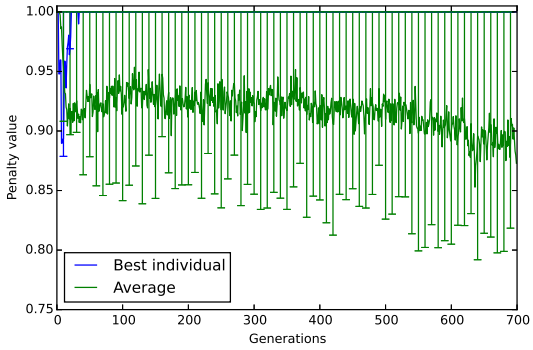
(d) Tenengrad in ITS



(e) Tenengrad in uninspired search



(f) Penalty in ITS



(g) Penalty in uninspired search

Figure 4.36: Comparison of ITS using $I = 0.04$, $R = 0.4$ with uninspired search using Tenengrad as feature function with the penalty.

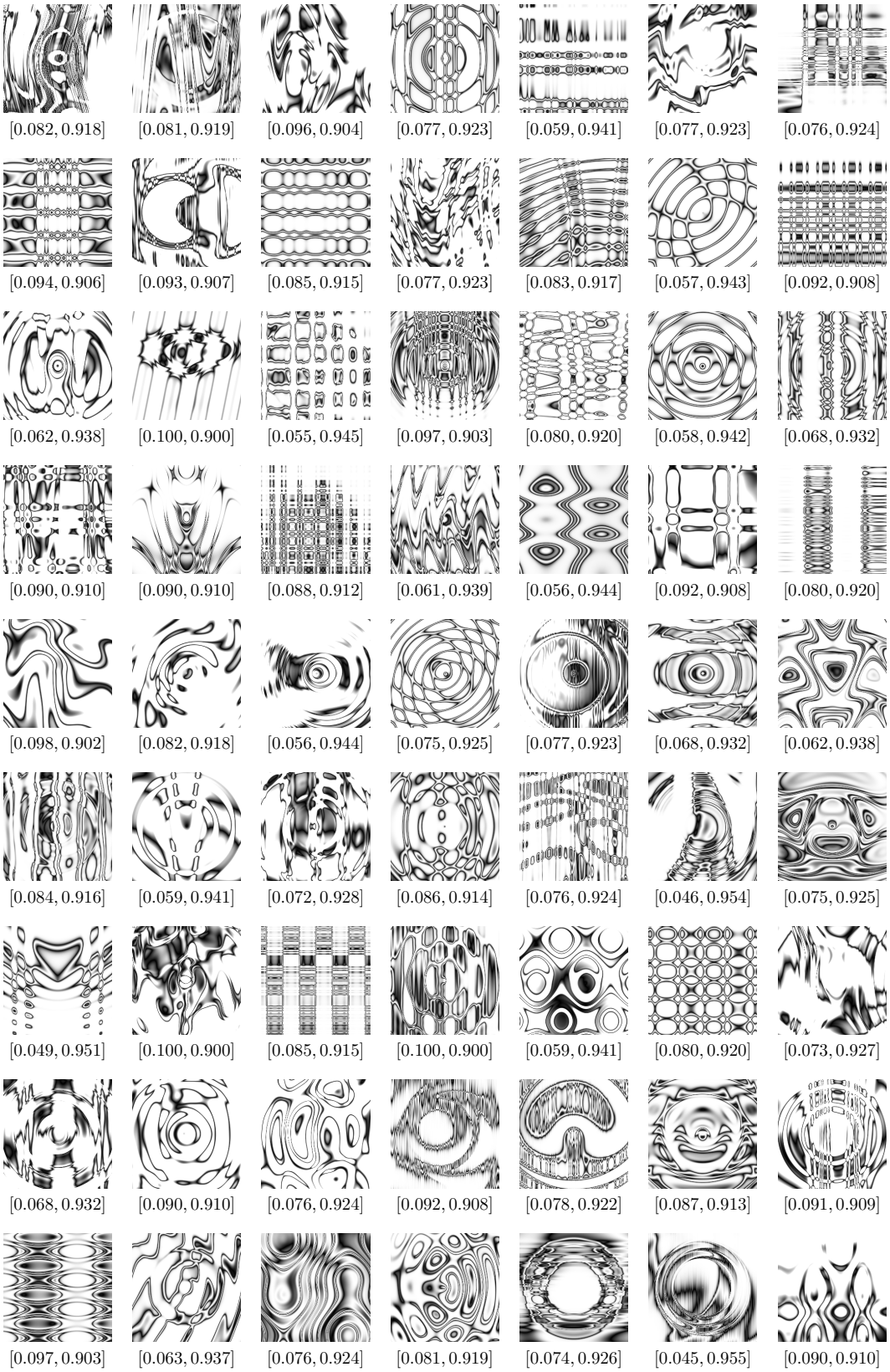


Figure 4.37: Examples of images found by ITS using Tenengrad as feature function with their complexity classifications according to the complexity classifier. The classification consists of the simplicity and the complexity node.

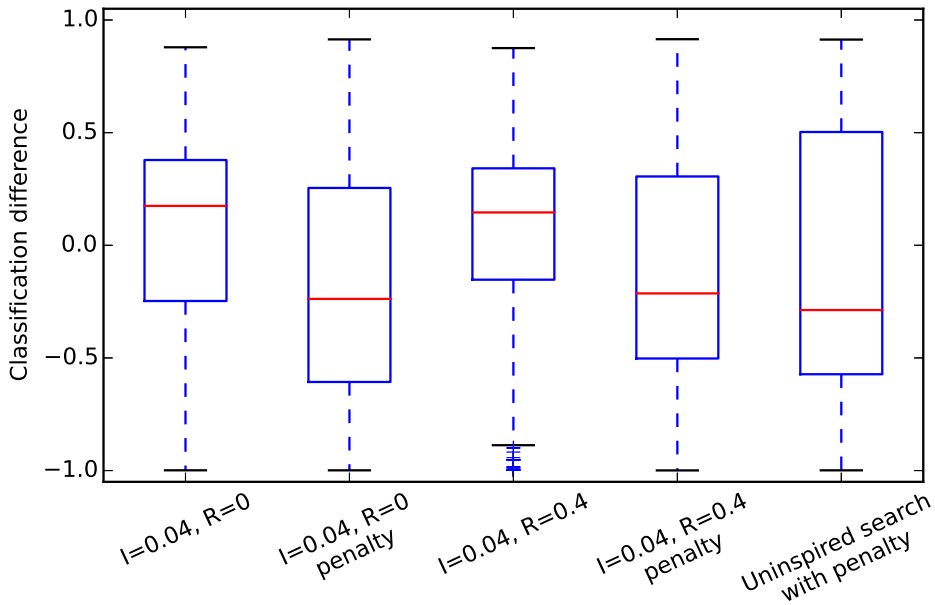


Figure 4.38: Comparison of ITS with different settings and with uninspired search by the complexity classifier Tenengrad as feature function.

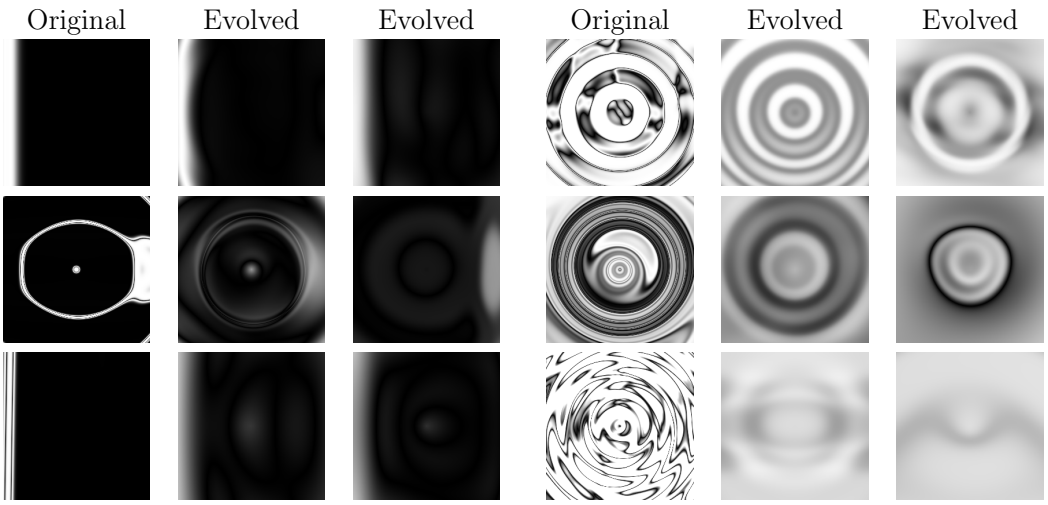
4.4.5 NV+RS

We tested uninspired search using normalized variance (NV), and relaxed symmetry (RS) as feature functions for 6 trials (one trial $\sim 94.0 \pm 50.7$ hours). The generated images were visually simple, see the figure 4.39 for examples. We selected 3 images from best individuals and used them for the complexity test by optimization in 3 trials. The result is 9 trials (one trial $\sim 49.9 \pm 14.5$ hours). The images were able to be found by the optimization, therefore images produced by uninspired search are classified as simple according to the complexity test by optimization, see the figure 4.41. The figure 4.40a shows examples of evolved images.

Figure 4.39: Examples of images found by uninspired search using normalized variance and relaxed symmetry as feature functions with their complexity classifications according to the complexity classifier.

We tested ITS using the inspiration threshold $I \in \{0.04, 0.05\}$, and the removal threshold $R \in \{0, 0.2, 0.3\}$. The figure 4.43a shows the influence of the particular setting on the total number of windows in the objective function. We tested each settings for 10 trials (1 trial $\sim 14.1 \pm 4.6$ hours). The figure 4.44 shows the diversity of evolved images.

We selected 3 images from all trials and used them for the complexity test by



(a) Uninspired search (b) ITS

Figure 4.40: Examples of images used for the complexity test by optimization. Shown evolved images are visually the most similar to the original image.

optimization in 5 trials. The result is 15 trials (one trial $\sim 52.2 \pm 14.7$ hours). The images were not able to be found by the optimization, therefore images produced by ITS are classified as complex according to the complexity test by optimization, see the figure 4.41. The figure 4.40b shows examples of evolved images. We can compare ITS with uninspired search by the best individuals from last generations, see the figure 4.41. ITS produced significantly more complex images ($p < 0.001$) according to the complexity test by optimization.

We can compare the different settings by the complexity classifier, see the figure 4.42. The removal of a window using $I = 0.04$ produced significantly more complex images ($p < 0.001$ for each combination). On the contrary, the removal of a window using $I = 0.05$ produced significantly less complex images ($p < 0.001$ for each combination). We selected the setting using $I = 0.04$, $R = 0.3$. ITS produced significantly more complex images ($p < 0.001$) according to the complexity classifier. Figures 4.43b, 4.43c compare the classification difference, the figures 4.43d, 4.43e and the figures 4.43f, 4.43g compare normalized variance and relaxed symmetry values of ITS with uninspired search over time.

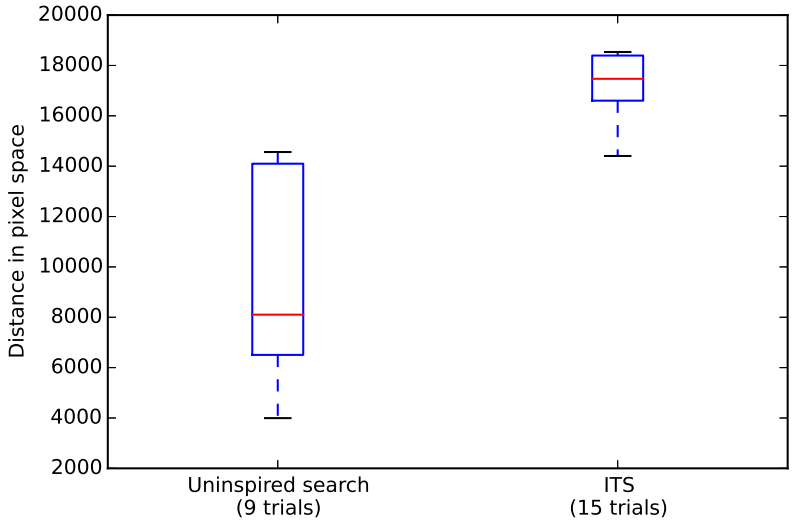


Figure 4.41: Comparison of ITS with uninspired search by the complexity test by optimization using normalized variance and relaxed symmetry as feature functions.

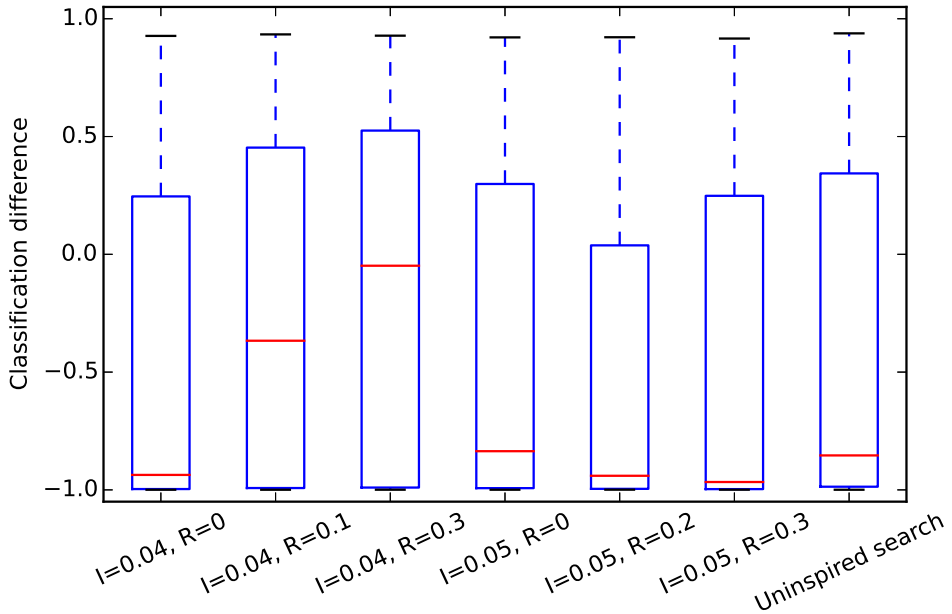
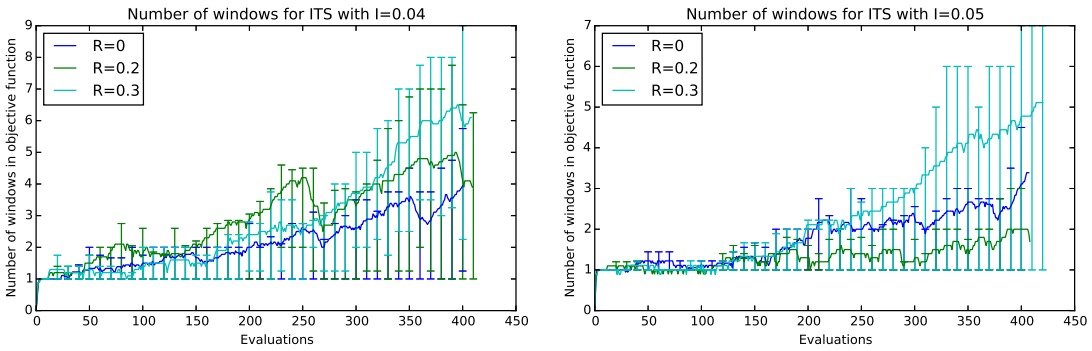
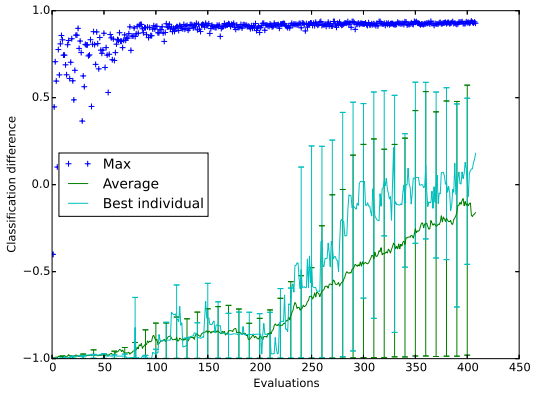


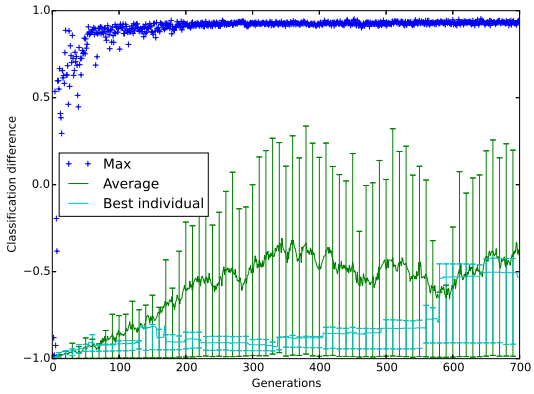
Figure 4.42: Comparison of ITS with different settings and with uninspired search by the complexity classifier using normalized variance and relaxed symmetry as feature functions.



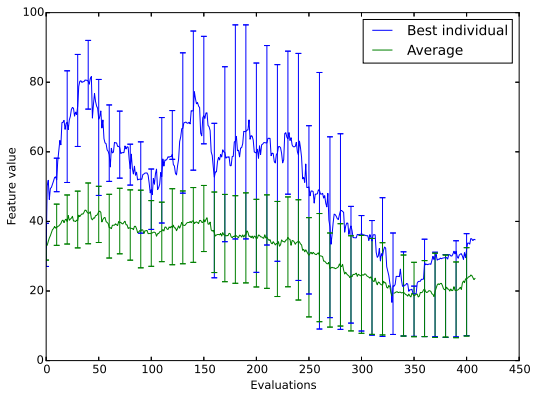
(a) Number of windows in the objective function for different settings in ITS.



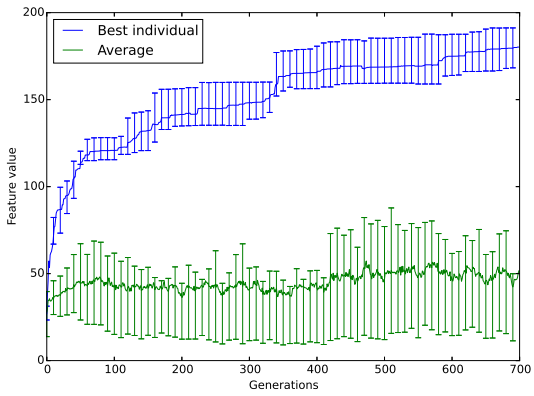
(b) Classification difference in ITS



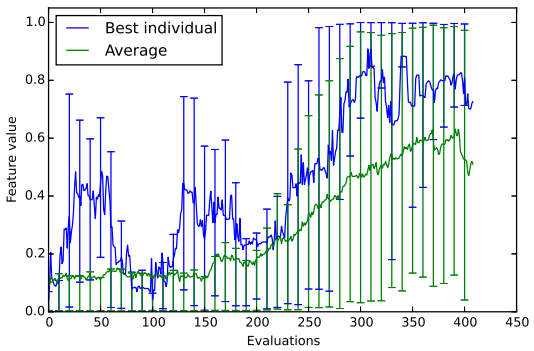
(c) Classification difference in uninspired search



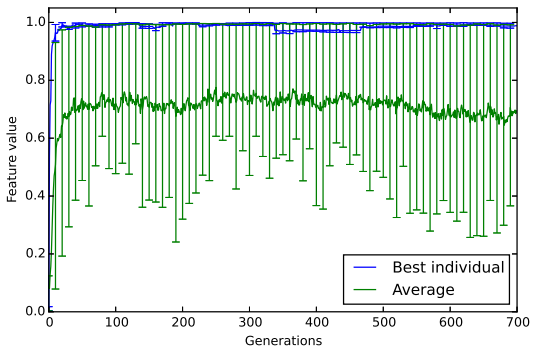
(d) Normalized variance in ITS



(e) Normalized variance in uninspired search



(f) Relaxed symmetry in ITS



(g) Relaxed symmetry in uninspired search

Figure 4.43: Comparison of ITS using $I = 0.04$, $R = 0.3$ with uninspired search using normalized variance and relaxed symmetry as feature functions.

Figure 4.44: Examples of images found by ITS using normalized variance and relaxed symmetry as feature functions with their complexity classifications according to the complexity classifier. The classification consists of the simplicity and the complexity node.

4.4.6 NV+RS+T

We tested uninspired search using normalized variance (NV), relaxed symmetry (RS), and Tenengrad (T) as feature functions with the penalty for 7 trials (one trial $\sim 131.3 \pm 46.6$ hours). The generated images were visually simple, see the figure 4.45 for examples. We selected 3 images from best individuals and used them for the complexity test by optimization in 3 trials. The result is 9 trials (one trial $\sim 49.9 \pm 14.5$ hours). The images were not able to be found by the optimization, therefore images produced by uninspired search are classified as complex according to the complexity test by optimization, see the figure 4.46b. The figure 4.46a shows examples of evolved images.

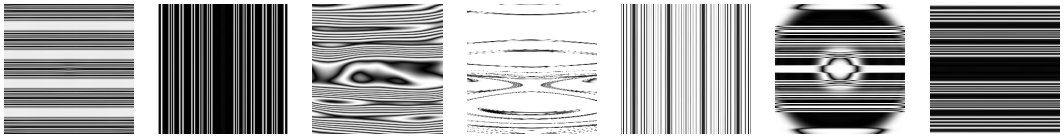
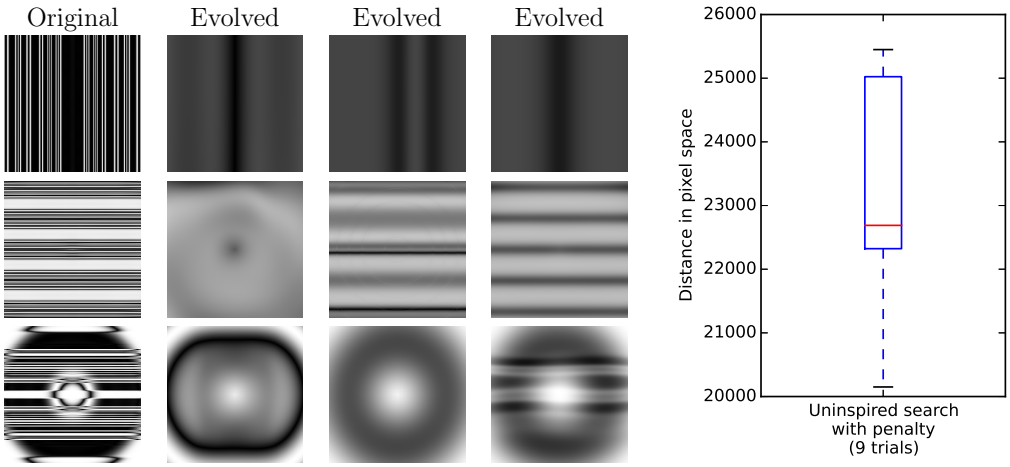


Figure 4.45: Examples of images found by uninspired search using normalized variance, relaxed symmetry, and Tenengrad as feature functions with the penalty with their complexity classifications according to the complexity classifier.



(a) Images used for the complexity test

(b) Complexity test

Figure 4.46: Uninspired search in the complexity test by optimization using normalized variance, relaxed symmetry, and Tenengrad as feature functions with the penalty.

We tested ITS using the inspiration threshold $I \in \{0.04, 0.1\}$, the removal threshold $R \in \{0, 0.4, 0.5\}$, and with and without the penalty. We tested each settings for 10 trials (1 trial $\sim 15.7 \pm 5.8$ hours). The figure 4.50 shows the diversity of evolved images.

We did not test ITS by the complexity test by optimization. We believe that images produced by ITS would be classified as complex, therefore we did not spend computational power on it. This belief is based on the results from ITS using only one feature function.

We selected the setting using $I = 0.1$, $R = 0.5$ and with the penalty in order to compare it with uninspired search which also used the penalty. Uninspired search produced more complex images ($p = 0.017$) according to the complexity classifier, see the figure 4.47. Nevertheless, we find images from ITS visually more interesting, compare the figure 4.50 with the figure 4.45. The figure 4.48 compares their classification difference and the figure 4.49 compares their features and penalty values over time.

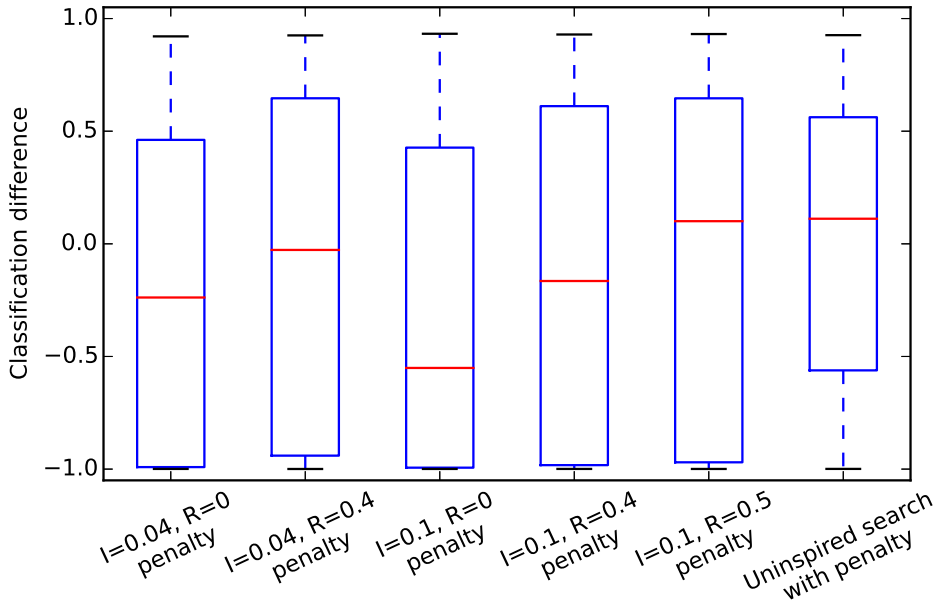
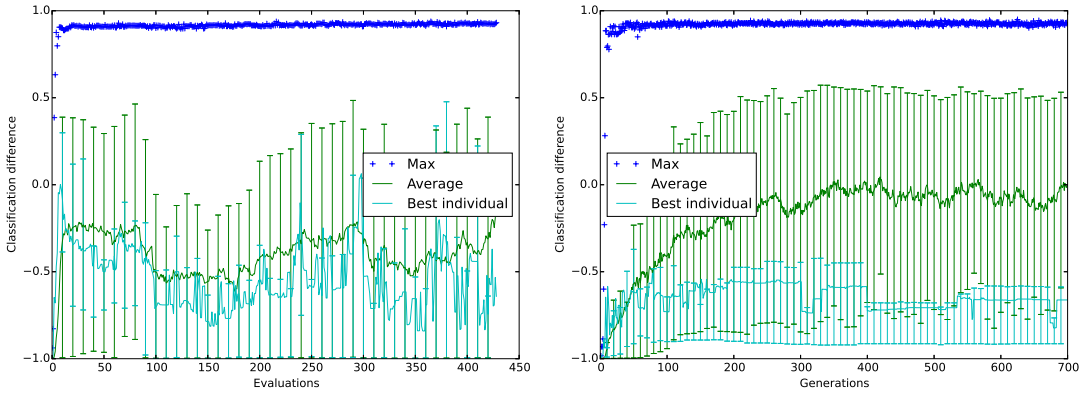
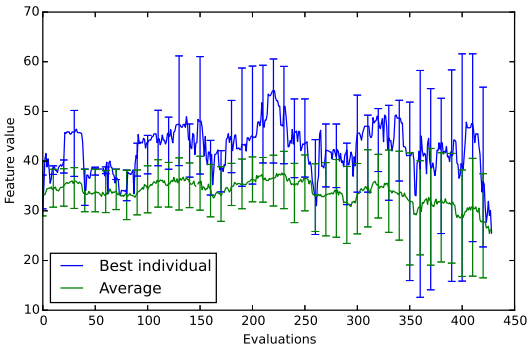


Figure 4.47: Comparison of ITS with different settings and with uninspired search by the complexity classifier using normalized variance, relaxed symmetry, and Tenengrad as feature functions.

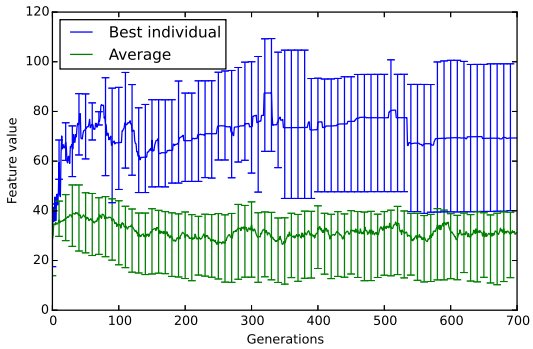


(a) Classification difference in ITS (b) Classification difference in uninspired search

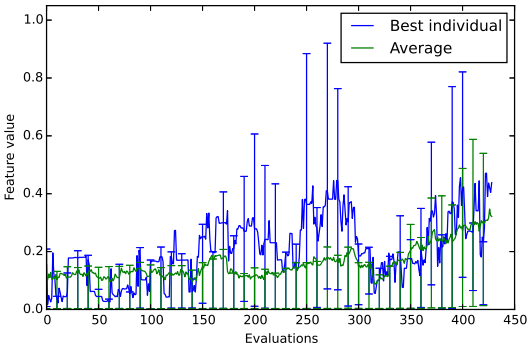
Figure 4.48: Comparison of ITS using $I = 0.1$, $R = 0.5$ with uninspired search using normalized variance, relaxed symmetry, and Tenengrad as feature functions with the penalty.



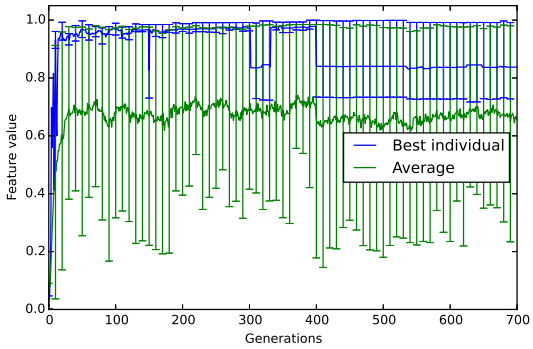
(a) Normalized variance in ITS



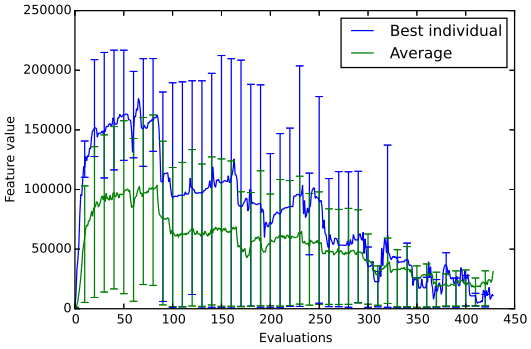
(b) Normalized variance in uninspired search



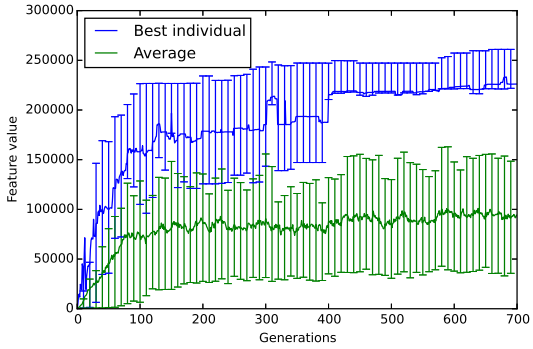
(c) Relaxed symmetry in ITS



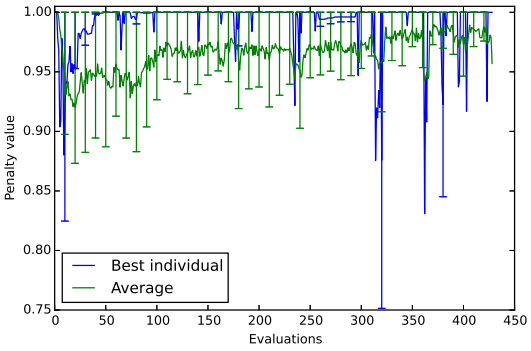
(d) Relaxed symmetry in uninspired search



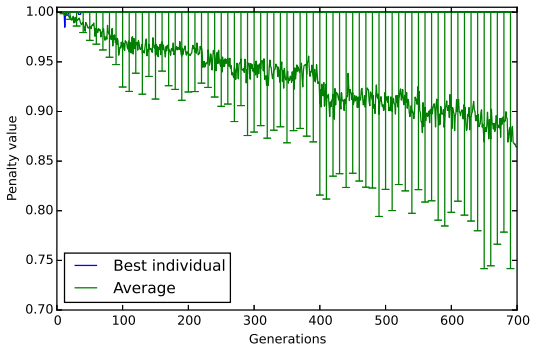
(e) Tenengrad in ITS



(f) Tenengrad in uninspired search



(g) Penalty in ITS



(h) Penalty in uninspired search

Figure 4.49: Comparison of ITS using $I = 0.1$, $R = 0.5$ with uninspired search using normalized variance, relaxed symmetry, and Tenengrad as feature functions with the penalty.

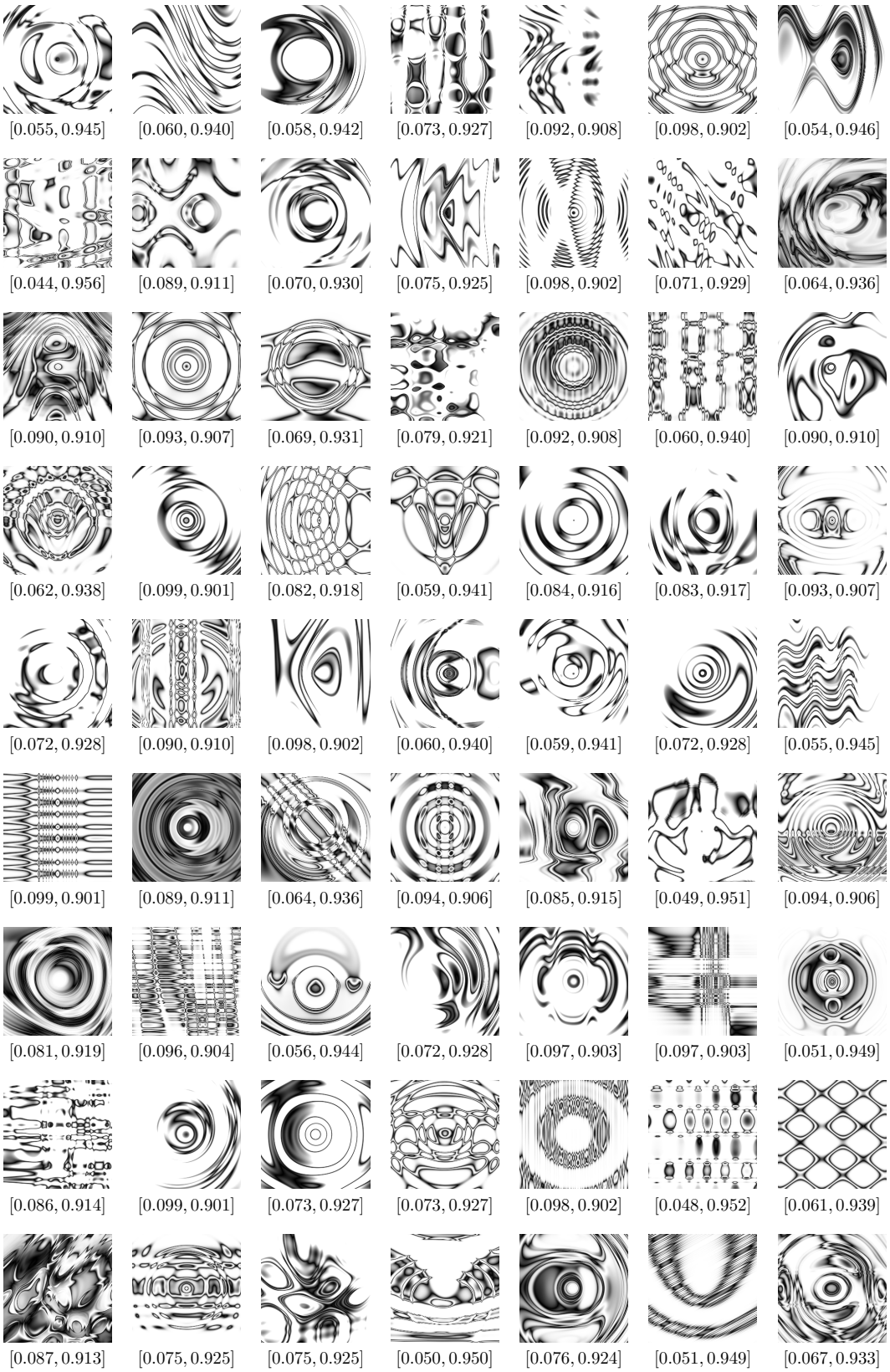


Figure 4.50: Examples of images found by ITS using normalized variance, relaxed symmetry, and Tenengrad as feature functions with their complexity classifications according to the complexity classifier. The classification consists of the simplicity and the complexity node.

4.4.7 NV+RS+GCF

We tested uninspired search using normalized variance (NV), relaxed symmetry (RS), and global contrast factor (GCF) as feature functions with the penalty for 3 trials (one trial $\sim 215.5 \pm 21.6$ hours). The generated images were visually simple, see the figure 4.51 for examples. We selected 3 images from best individuals and used them for the complexity test by optimization in 3 trials. The result is 9 trials (one trial $\sim 53.7 \pm 10.6$ hours). The images were not able to be found by the optimization, therefore images produced by uninspired search are classified as complex according to the complexity test by optimization, see the figure 4.52b. The figure 4.52a shows examples of evolved images.

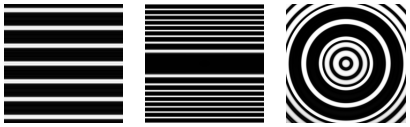
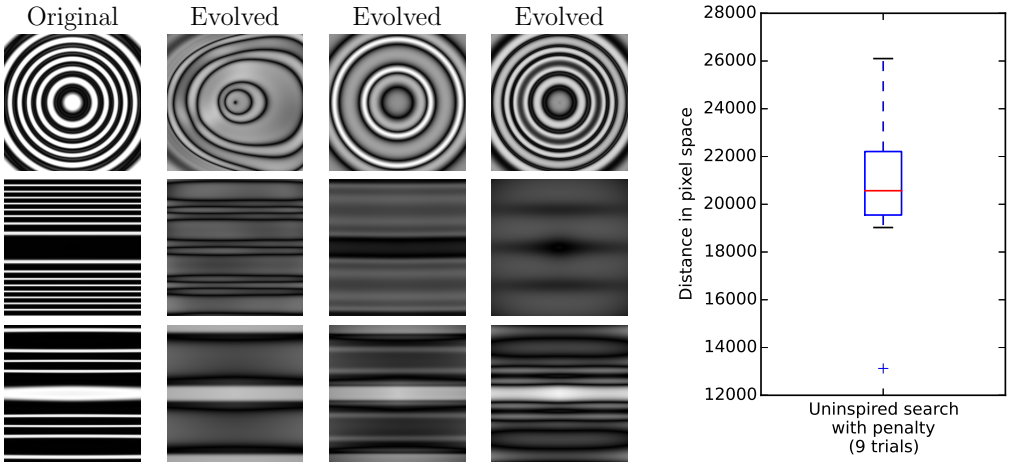


Figure 4.51: Examples of images found by uninspired search using normalized variance, relaxed symmetry, and global contrast factor as feature functions with the penalty with their complexity classifications according to the complexity classifier.



(a) Images used for the complexity test

(b) Complexity test

Figure 4.52: Uninspired search in the complexity test by optimization using normalized variance, relaxed symmetry, and global contrast factor as feature functions with the penalty.

We tested ITS using the inspiration threshold $I \in \{0.04, 0.1\}$, the removal threshold $R \in \{0, 0.4, 0.5\}$, and with and without the penalty. We tested each settings for 10 trials (1 trial $\sim 16.7 \pm 9.7$ hours). The figure 4.56 shows the diversity of evolved images.

We did not test ITS by the complexity test by optimization. We believe that images produced by ITS would be classified as complex, therefore we did not spend computational power on it. This belief is based on the results from ITS using only one feature function.

We selected the setting using $I = 0.04$, $R = 0$ and with the penalty in order to compare it with uninspired search which also used the penalty. Uninspired search produced significantly more complex images ($p < 0.001$ for any combination) according to the complexity classifier, see the figure 4.53. Nevertheless, we find images from ITS visually more interesting, compare the figure 4.56 with the figure 4.51. The figure 4.54 compares their classification difference and the figure 4.55 compares their features and penalty values over time.

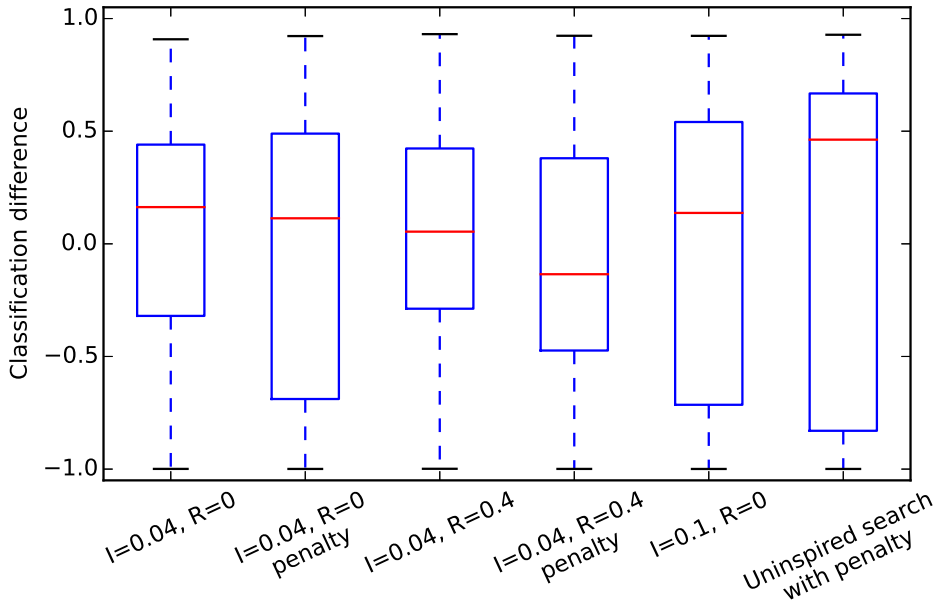
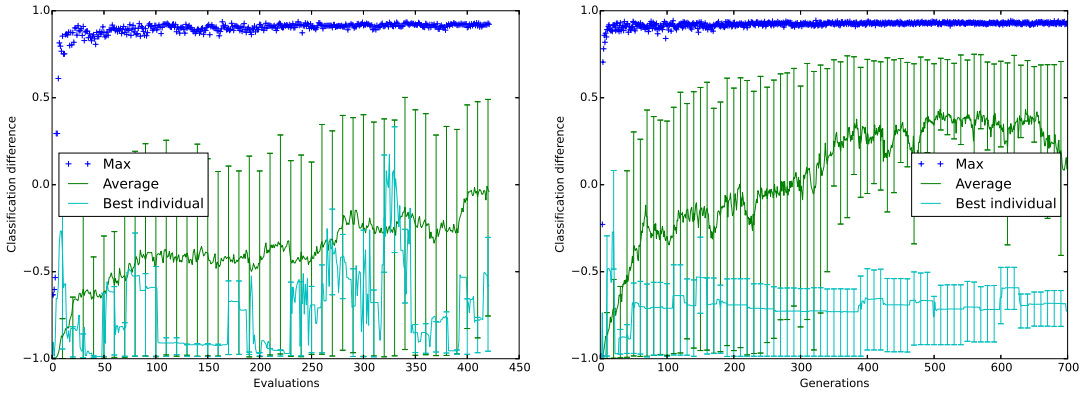
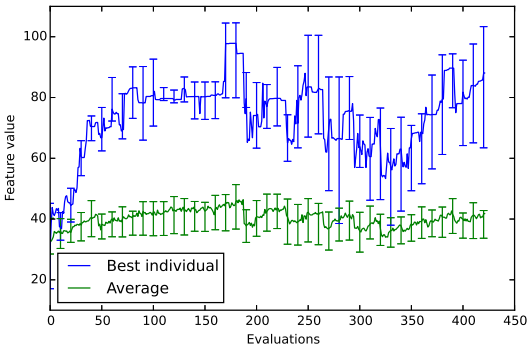


Figure 4.53: Comparison of ITS with different settings and with uninspired search by the complexity classifier using normalized variance, relaxed symmetry, and global contrast factor as feature functions.

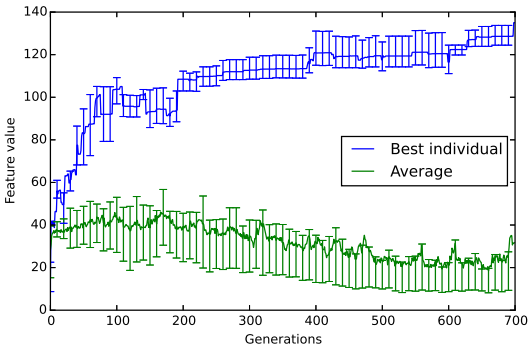


(a) Classification difference in ITS (b) Classification difference in uninspired search

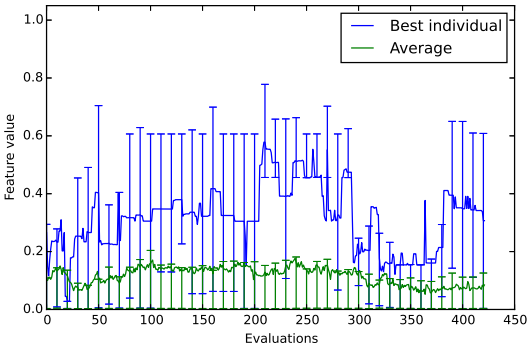
Figure 4.54: Comparison of ITS using $I = 0.04$, $R = 0$ with uninspired search using normalized variance, relaxed symmetry, and global contrast factor as feature functions with the penalty.



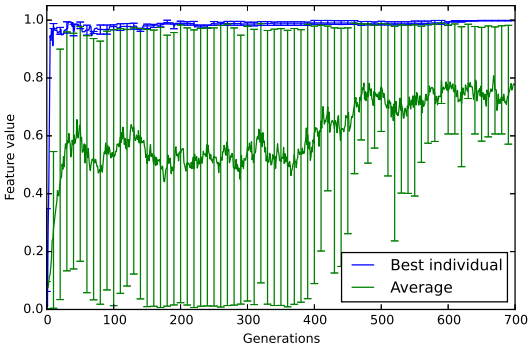
(a) Normalized variance in ITS



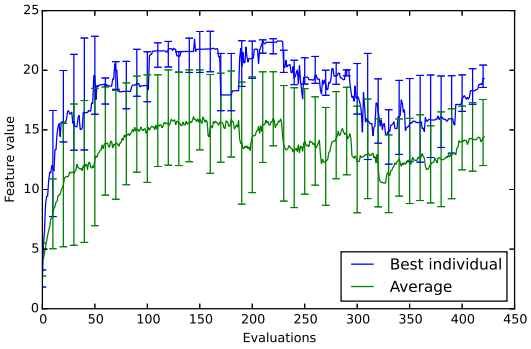
(b) Normalized variance in uninspired search



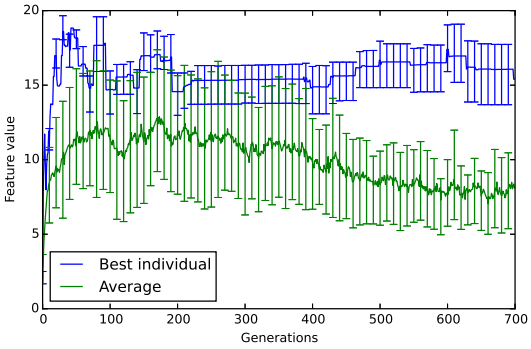
(c) Relaxed symmetry in ITS



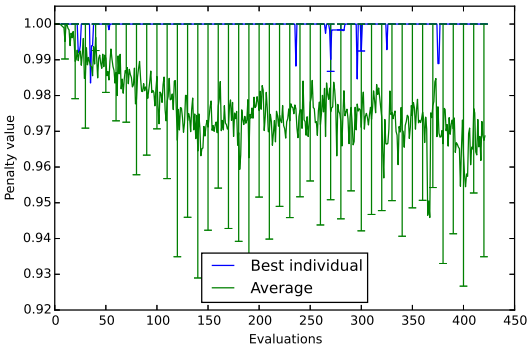
(d) Relaxed symmetry in uninspired search



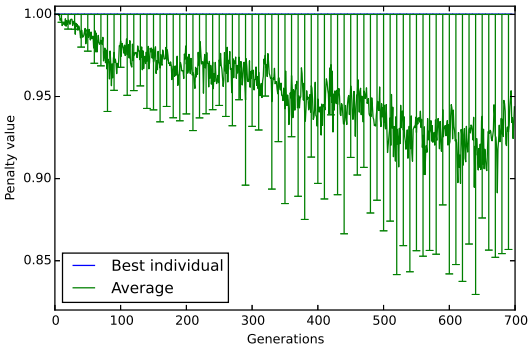
(e) Global contrast factor in ITS



(f) Global contrast factor in uninspired search



(g) Penalty in ITS



(h) Penalty in uninspired search

Figure 4.55: Comparison of ITS using $I = 0.04$, $R = 0$ with uninspired search using normalized variance, relaxed symmetry, and global contrast factor as feature functions with the penalty.

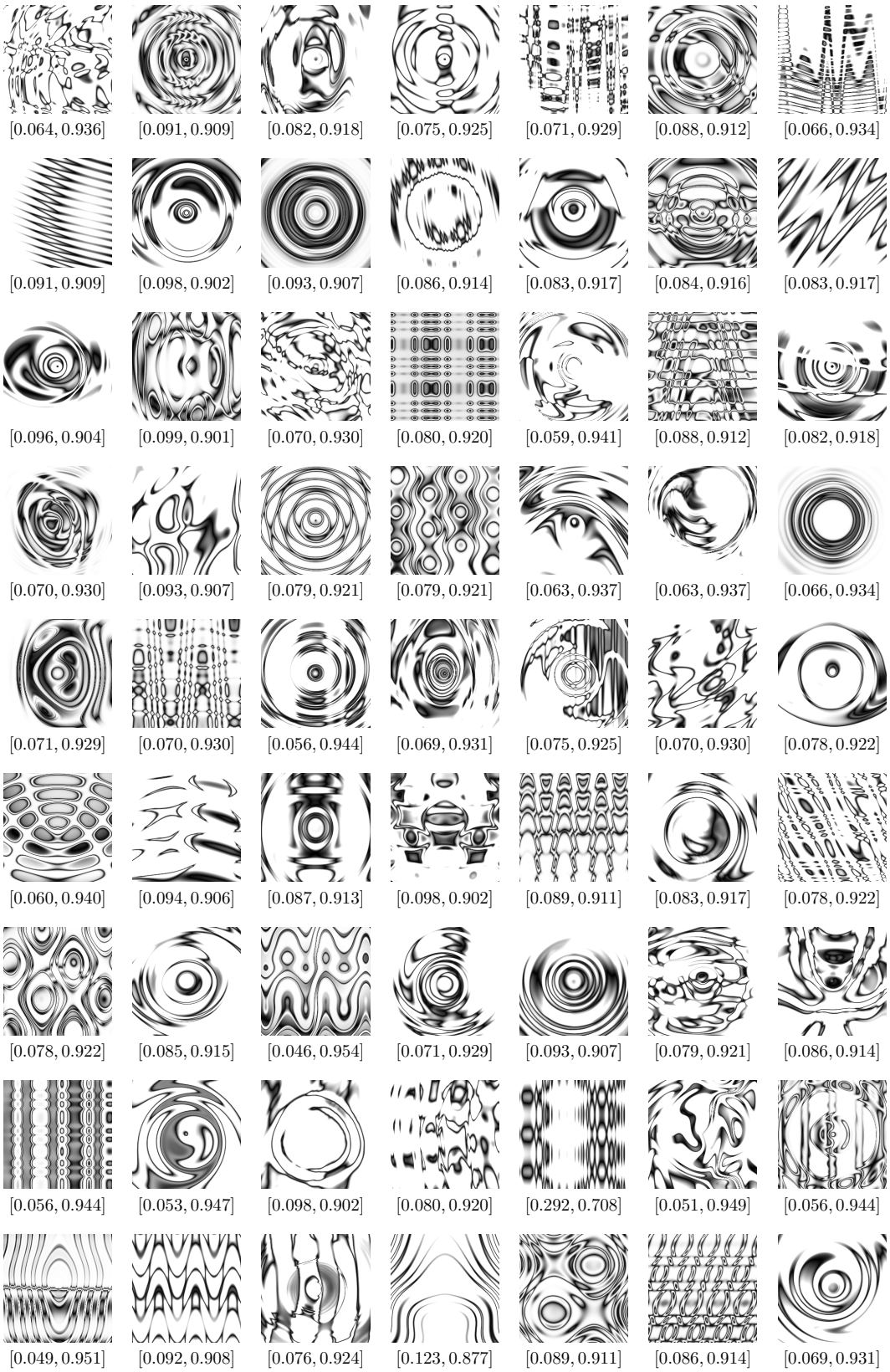
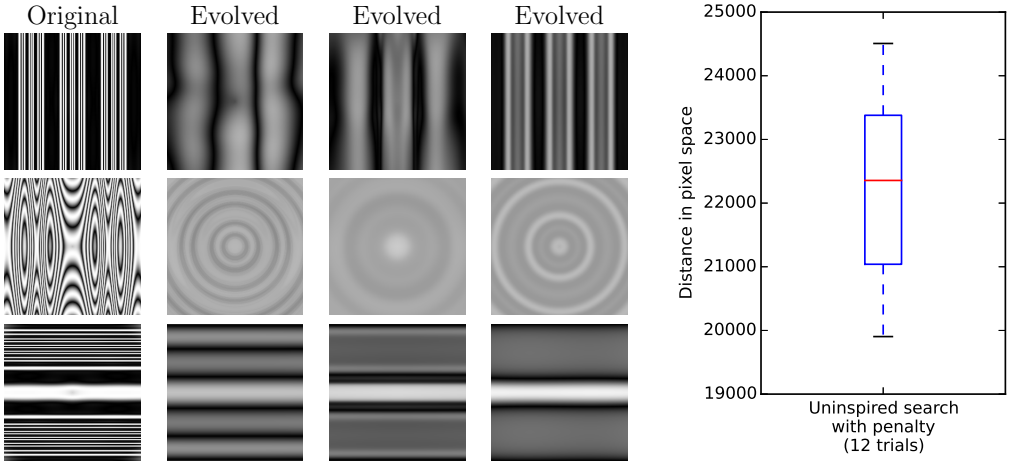


Figure 4.56: Examples of images found by ITS using normalized variance, relaxed symmetry, and global contrast factor as feature functions with their complexity classifications according to the complexity classifier. The classification consists of the simplicity and the complexity node.

4.4.8 NV+RS+GCF+T

We tested uninspired search using normalized variance (NV), relaxed symmetry (RS), global contrast factor (GCF), and Tenengrad (T) as feature functions with the penalty for 5 trials (one trial $\sim 122.3 \pm 21.0$ hours). The generated images were visually simple, see the figure 4.57 for examples. We selected 4 images from best individuals and used them for the complexity test by optimization in 3 trials. The result is 12 trials (one trial $\sim 50.1 \pm 10.9$ hours). The images were not able to be found by the optimization, therefore images produced by uninspired search are classified as complex according to the complexity test by optimization, see the figure 4.58b. The figure 4.58a shows examples of evolved images.

Figure 4.57: Examples of images found by uninspired search using normalized variance, relaxed symmetry, global contrast factor, and Tenengrad as feature functions with the penalty with their complexity classifications according to the complexity classifier.



(a) Images used for the complexity test

(b) Complexity test

Figure 4.58: Uninspired search in the complexity test by optimization using normalized variance, relaxed symmetry, global contrast factor, and Tenengrad as feature functions with the penalty.

We tested ITS using the inspiration threshold $I \in \{0.04, 0.1\}$, the removal threshold $R \in \{0, 0.4, 0.5\}$, and with and without the penalty. We tested each settings for 10 trials (1 trial $\sim 17.9 \pm 8.2$ hours). The figure 4.62 shows the diversity of evolved images.

We did not test ITS by the complexity test by optimization. We believe that images produced by ITS would be classified as complex, therefore we did not

spend computational power on it. This belief is based on the results from ITS using only one feature function.

We selected the setting using $I = 0.1$, $R = 0.5$ and with the penalty in order to compare it with uninspired search which also used the penalty. Uninspired search produced significantly more complex images ($p < 0.001$ for any combination) according to the complexity classifier, see the figure 4.59. Nevertheless, we find images from ITS visually more interesting, compare the figure 4.62 with the figure 4.57. The figure 4.60 compares their classification difference and the figure 4.61 compares their features values over time.

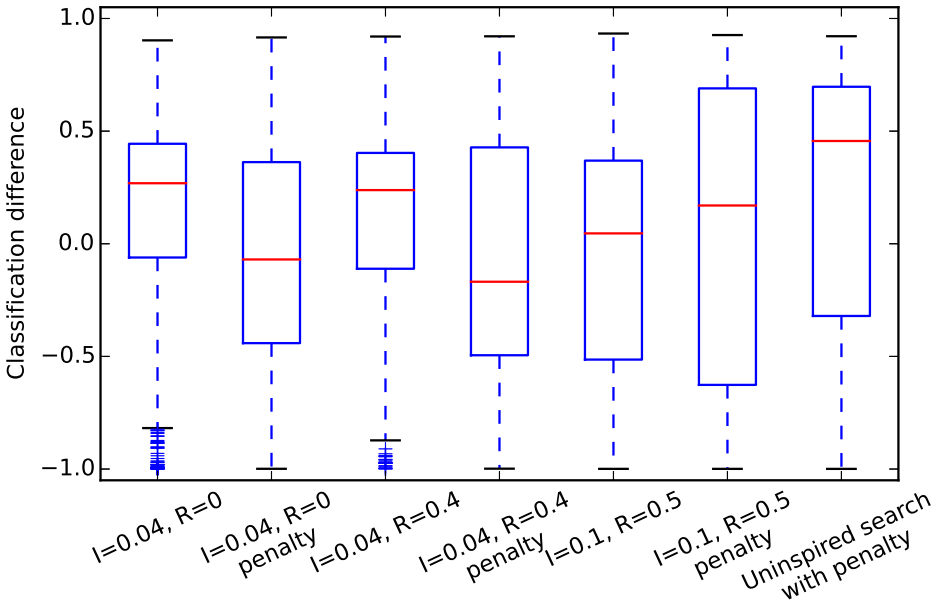
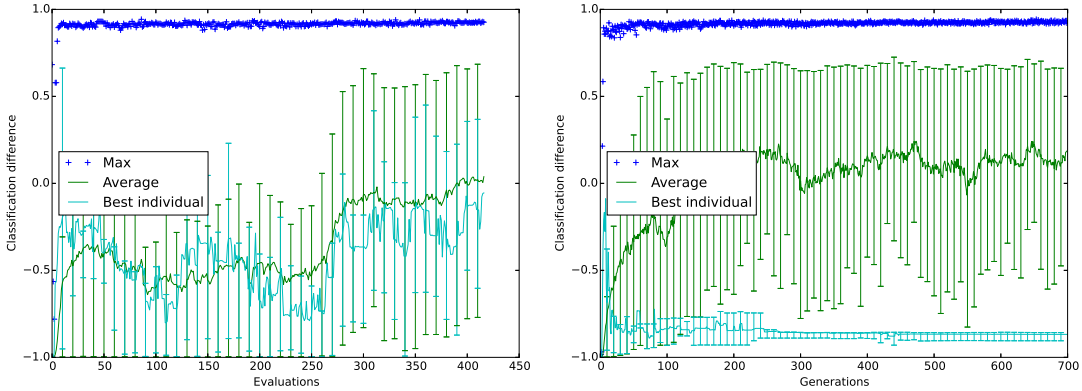
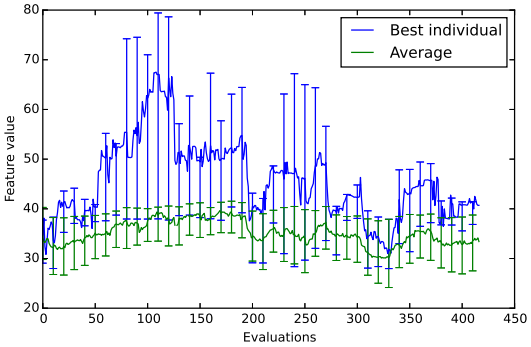


Figure 4.59: Comparison of ITS with different settings and with uninspired search by the complexity classifier using normalized variance, relaxed symmetry, global contrast factor, and Tenengrad as feature functions.

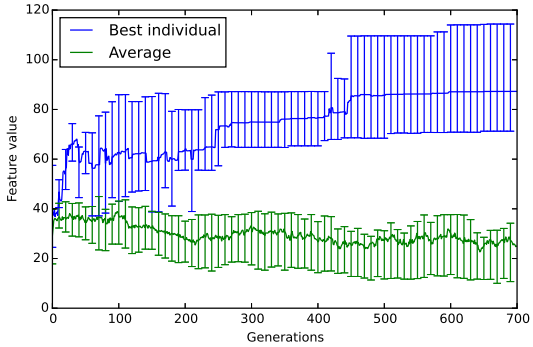


(a) Classification difference in ITS (b) Classification difference in uninspired search

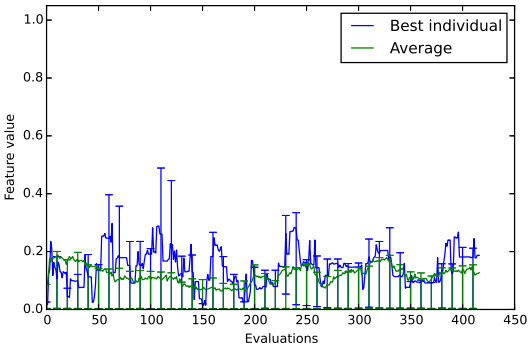
Figure 4.60: Comparison of ITS using $I = 0.1$, $R = 0.5$ with uninspired search using normalized variance, relaxed symmetry, global contrast factor, and Tenengrad as feature functions with the penalty.



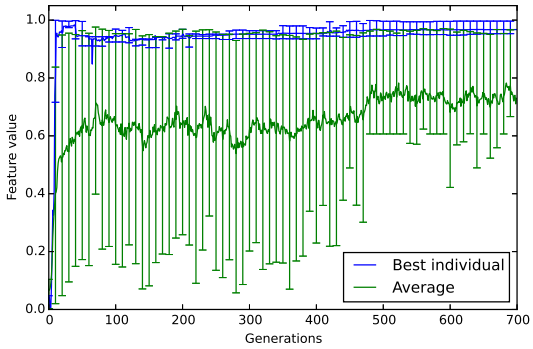
(a) Normalized variance in ITS



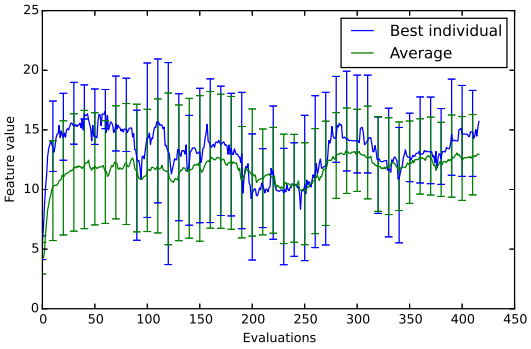
(b) Normalized variance in uninspired search



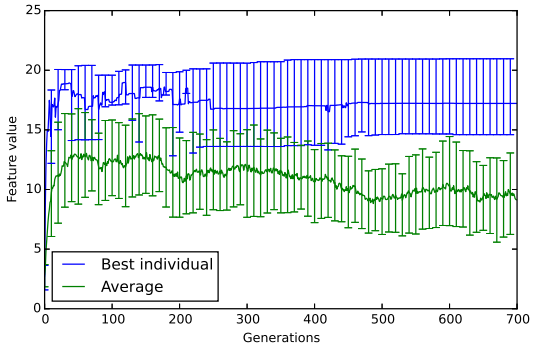
(c) Relaxed symmetry in ITS



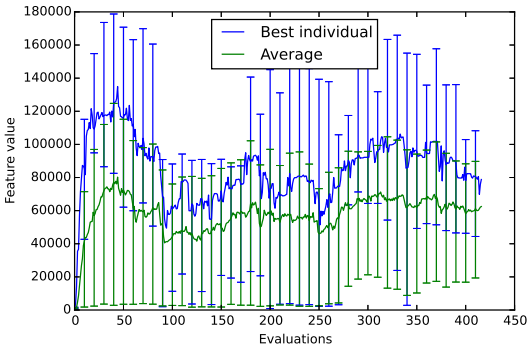
(d) Relaxed symmetry in uninspired search



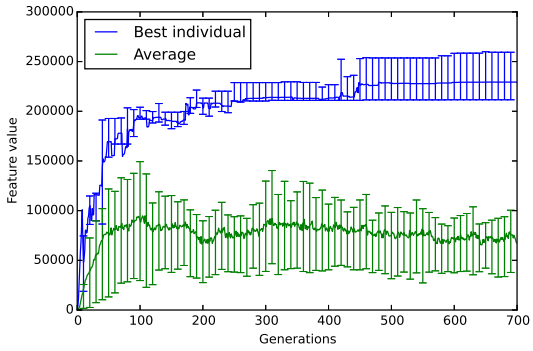
(e) Global contrast factor in ITS



(f) Global contrast factor in uninspired search



(g) Tenengrad in ITS



(h) Tenengrad in uninspired search

Figure 4.61: Comparison of ITS using $I = 0.1$, $R = 0.5$ with uninspired search using normalized variance, relaxed symmetry, global contrast factor, and Tenengrad as feature functions with the penalty.

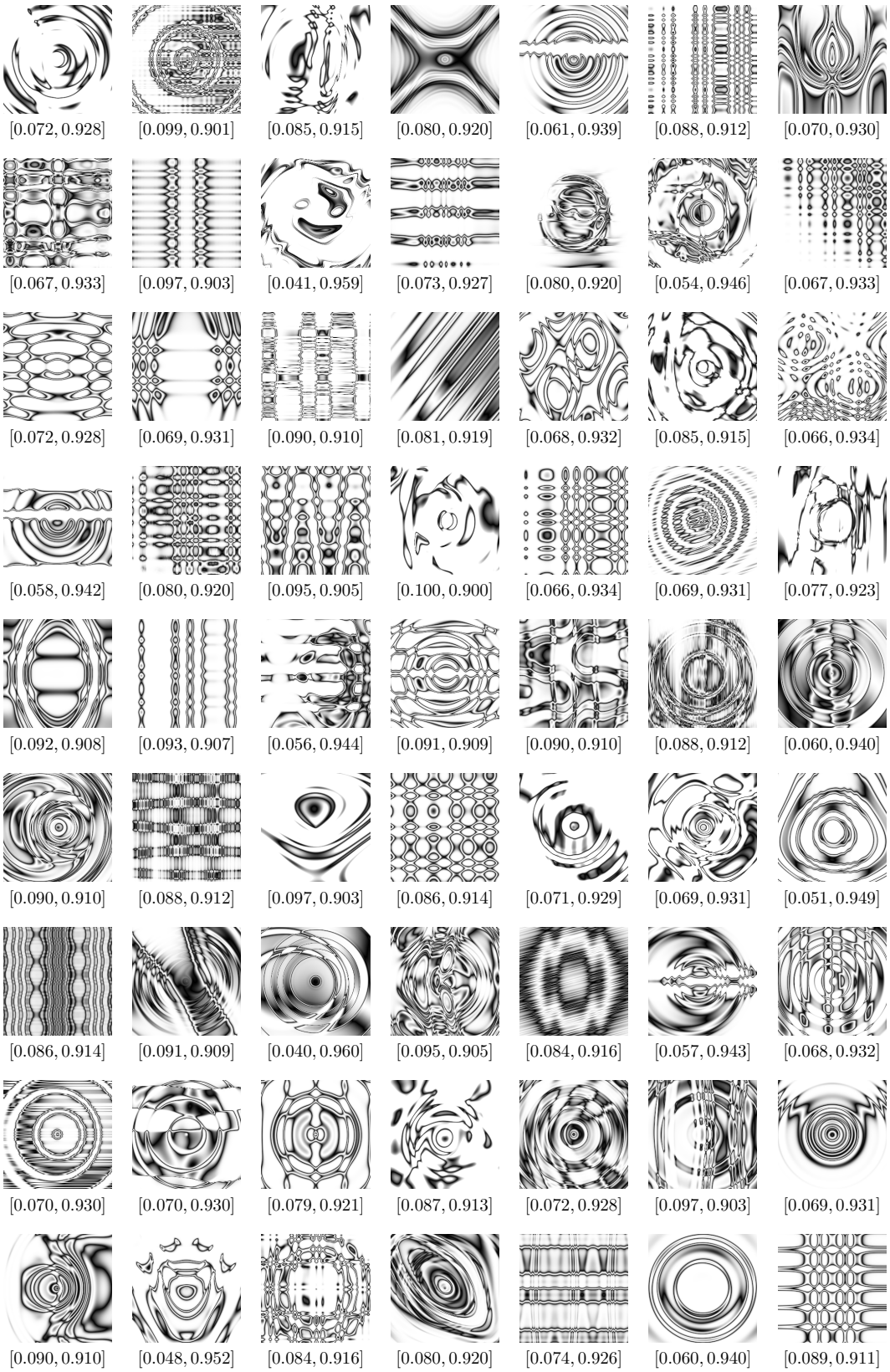


Figure 4.62: Examples of images found by ITS using normalized variance, relaxed symmetry, global contrast factor, and Tenengrad as feature functions with their complexity classifications according to the complexity classifier. The classification consists of the simplicity and the complexity node.

4.5 Summary

In this section, we summarize the results of our evaluation. We used two methods to compare ITS with uninspired search using the same set of feature functions: the complexity test by optimization and the complexity classifier. We proposed these methods to have a non-subjective measure. However, we also showed their limitations during our evaluation. Additionally, we leave a visual comparison up to the reader.

The table 4.1 shows a summary according to the complexity test by optimization. The complexity of images from uninspired search is biased by the particular feature function. Normalized variance, and relaxed symmetry produced simple images, even when they were used together. On the contrary, global contrast factor, and Tenengrad produced complex images according to the complexity test by optimization. When the set of features for uninspired search contained any of the latter, the resulting images were classified as complex. We think that these images are mainly from the class II or IV. We showed that the complexity test classifies these classes as complex. When both algorithms produced images classified as complex, the statistical comparison is biased by the chosen images for the complexity test.

We did not use the complexity test for the multiple features with global contrast factor, or Tenengrad. Nevertheless, we believe that images produced by ITS would be classified as complex and uninspired search would statistically produce more complex images. We base this idea on the results from using global contrast, or Tenengrad as a single feature function.

Set of features	Uninspired search	Comparison	ITS	p-value
NV	simple	$<$	complex	< 0.001
RS	simple	$<$	complex	< 0.001
GCF	complex	$>$	complex	< 0.001
T	complex	$>$	complex	< 0.001
NV+RS	simple	$<$	complex	< 0.001
NV+RS+T	complex		$\times$	
NV+RS+GCF	complex		$\times$	
NV+RS+GCF+T	complex		$\times$	

Table 4.1: Summary of the evaluation according to the complexity test by optimization. Images from the particular algorithm are classified by the complexity test. Results of comparisons are shown with corresponding p-values. NV is normalized variance, RS is relaxed symmetry, GCF is global contrast factor, and T is Tenengrad.

The table 4.2 shows a summary according to the complexity classifier. Similarly to the complexity test by optimization, the complexity of images from uninspired search is biased by the particular feature function. When the set of features for uninspired search contains global contrast factor or Tenengrad, the resulting images are more likely to be classified as complex. We think it is caused by our data set to train a classifier. We labelled an image as complex, when it was sharp. Images created from the latter features are mainly sharp, therefore

they are classified as complex according to the complexity classifier. On the other hand, they are visually very simple according to our subjective comparison. The problem could also be caused by the unfair aggregation of all trials by evaluations, because each run of ITS is unique.

Set of features	Uninspired search	Comparison	ITS	p-value
NV	−0.986	$<$	−0.335	< 0.001
RS	0.353	$<$	0.532	< 0.001
GCF	−0.112	$\approx$	0.027	0.164
T	−0.288	$\approx$	−0.213	0.491
NV+RS	−0.853	$<$	−0.048	< 0.001
NV+RS+T	0.111	$>$	0.101	0.017
NV+RS+GCF	0.462	$>$	0.113	< 0.001
NV+RS+GCF+T	0.456	$>$	0.170	< 0.001

Table 4.2: Summary of the evaluation according to the complexity classifier. Each algorithm shows a median of complexity difference of produced images from all trials. Results of comparisons are shown with corresponding p-values. NV is normalized variance, RS is relaxed symmetry, GCF is global contrast factor, and T is Tenengrad.

Conclusion

In this thesis, we investigated a different approach to deal with the trap of local optima in stochastic optimization methods from machine learning. Our approach was inspired by the users interacting with Picbreeder. Our hypothesis was that their behaviours depict creative processes. These processes have no fixed goals. We introduced a general framework on the top of a common optimization technique called inspiration-triggered search (ITS), which mimics these processes. We tested our approach in the domain of images, that is to find "complex" and aesthetically pleasant images for humans.

ITS generalizes the behaviour into three phases: the inspiration phase, the optimization phase, and the diversification phase. In the inspiration phase, the population in the optimization technique is analysed to detect "interesting or promising" properties and a new objective function is defined. The optimization phase represents the standard optimization using the current objective function. The idea of the diversification phase is to add diversity to the population after the convergence of the optimization technique. ITS diversifies the population until something "inspiring" in the population is found. ITS can be considered as an automated incremental approach. It can start with a simple objective function and continuously make it more complex. For this reason, the current population is crucial. It contains individuals that do not perform poorly for the new defined objective function, otherwise the search would have to deal with the bootstrap problem.

We introduced a general idea of descriptions of individuals that describe their "good" properties. We proposed one instance in which descriptions represent "promising" parts of individuals, that we call windows. In other words, the original problem is split into sub-problems defined by windows. In our testing domain of images, we defined windows by a detection of contours. The objective function uses the same idea of windows. It is defined by a collection of windows for each feature and the optimization technique focuses on these windows. If an "inspiring" individual appears in the population, the objective function will be modified by the individual's description. We defined inspiring individuals by high values of a user-defined description metric between the objective function and the description of the particular individual. We proposed two basic operators to modify the objective function: adding a new window to the objective function, and removing an existing window from the objective function.

In our experiments, images were represented by CPPNs and we used NEAT as the underlying optimization technique. We tested measures from the field of computational aesthetics and measures defining a sharpness of the image. We subjectively chose measures as features for the optimization that produced some visual results.

We compared ITS with the underlying optimization technique using the same set of features. The main problem was the definition of a complex image. Our first approach was to test whether the images can be found or not by the used optimization technique using the image as the objective function. We defined that the image is complex when it cannot be found. However, we showed that even visually simple images are not able to be found by this approach. Secondly,

we used machine learning methods to train a classifier to distinguish "simple" and "complex" images. The problem was that sometimes we were not able to tell whether images are complex or not. We labelled images as complex, when they were sharp or contained more shapes. Nevertheless, some features produced visually simple but sharp images, therefore the images were classified as complex. This was a problem for our comparison. Even though, the produced images were visually simple, they had high complexities according to the used classifier. Unfortunately, none of these two methods gave us a conclusive answer whether our proposed method produces more "complex" images or not. Nevertheless, we found images produced by our approach that are visually and subjectively interesting.

The resulting "complexity" of images is biased by the used feature function. The direct optimization of normalized variance or relaxed symmetry produced visually very simple images. For them, we were able to show that our approach produced more "complex" images according to both of proposed methods. The direct optimization of global contrast factor and Tenengrad still produced visually simple images according to our subjective perspective. However, the produced images were more complex in comparison with images produced by normalized variance or relaxed symmetry. For them, both of used methods were not sufficient and the results were not conclusive.

The study of complexibility is required for a deeper evaluation of our proposed approach. This problem of "subjective complexity" is not only connected with the domain of images. For instance, in the field of evolutionary robotics, let us imagine we want to evolve a neural network that would allow a mobile robot to move to a specified target. Fitness functions based only on times to get there and lengths of trajectories do not contain the whole information about "complexities" of evolved behaviours. For instance, shapes of trajectories are important as well. However, this information is difficult to formalize. For this reason, human operators are usually used to rate these shapes of trajectories. Such a view does not give us much hope: if evolving simple behaviours is not straightforward, how could we hope to employ evolution to design vastly more complex and hard to define behaviours such as "being intelligent"?

On the other hand, we must expect that produced images are biased by the used set of features. We should not expect that we would be able to find The Mona Lisa painting by Leonardo da Vinci by only optimizing normalized variance. We would need at least a feature recognizing human figures. By including creative processes in the search, we mainly argue that the search is able to found more "complex" solutions in comparison with the direct optimization.

Future work

We would like to compare our proposed method with similar approaches that do not use fixed objective functions, for instance, such as novelty search.

We must more investigate the diversification phase to conclude whether our use helps in most cases or not. We only showed that it is a double-edged sword because it usually caused a significant temporal drop in complexity according to the complexity classifier. Also, we need to examine other definitions of operators to modify the objective function. For instance, we did not use an operator to

modify an existing window in the objective function. It could be convenient to change a size or a position of a particular window.

Next researches should focus on combing ITS with an objective-based search or improving an objective-based search by ITS.

Lastly, we briefly mention several ideas without concrete details that we did not use for the presented ITS in this thesis

Dynamic size Sizes of problems from a particular domain do not have to be fixed, for instance, resolutions of images. We could start with a minimal size and dynamically increase or decrease the problem's size according to the objective function. It would represent an open-ended evolution. For instance, in the domain of images, the size of the generated image can be defined by a margin from windows of the objective function. If a new added window to the objective function is close to borders, the image's resolution will be increased. Similarly, a removed window can decrease the image's resolution. The idea is that artists do not have to start with fixed sizes of their artworks. For instance, lengths of songs are not usually defined in advance but they are more likely results of particular ideas of music composers.

Features as properties We would like to use features for which we do not know which value is better. In other words, a feature value only describes a property of an individual. These features represent typical properties used by humans. Their values are usually incomparable. For instance, let us imagine a feature that describes usage of bass guitar in songs. Do "good" songs contain a bass guitar? If yes, how much should it be used? There is no clear answer. We could find "good" songs for almost any variant. This example illustrates that a simple maximization or minimization of some properties do not make sense.

For this type of features, we could extract some information from "good" individuals. A simple option would be to use an average feature value from these individuals and optimize towards this value in a particular window. Additionally, changes of the objective function could modify this value.

Multi-agent ITS One could consider that ITS evolves objective functions. In this thesis, ITS evolved only one objective function and it could be consider as one artist. However, we could use more artists and evolve more objective functions at the same time. It would not be the same as island models [55] used in evolutionary computation, in which separate evolutionary algorithms run independently on each island and interact by means of migrating individuals. We would rather migrate objective functions than individuals, or precisely, an objective function of one island could be used to modify objective functions of other islands. The main idea is to achieve a self-organization similar to art movements in human history. One artist finds something interesting and can influence other artists to some extent. This behaviour represents a multi-agent system. Even, Picbreeder used more users to create best rated images by branching. We can draw inspiration from approaches [69, 99, 102] used in evolutionary computation inspired by behaviours in human societies.

Bibliography

[1] Acebo, E ; Sbert, Mateu: Benford's law for natural and synthetic images. In: *Proceedings of the First Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging* Eurographics Association, 2005, S. 169–176

[2] Alpaydin, Ethem: *Introduction to Machine Learning*. MIT Press, 2014

[3] Auerbach, Joshua E. ; Bongard, Josh C.: Evolving complete robots with CPPN-NEAT: the utility of recurrent connections. In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation* ACM, 2011, S. 1475–1482

[4] Avnet, Jeremy: *Computation, dynamics and the phase-transition*. 2000

[5] Baluja, Shumeet ; Pomerleau, Dean ; Jochem, Todd: Towards automated artificial evolution for computer-generated images. In: *Connection Science* 6 (1994), Nr. 2-3, S. 325–354

[6] Bear, Mark F. ; Connors, Barry W. ; Paradiso, Michael A.: *Neuroscience: Exploring the brain*. 3rd. Lippincott Williams & Wilkins, 2007

[7] Bense, M: *Einführung in die informationstheoretische Ästhetik. Grundlegung und Anwendung in der Texttheorie (Introduction to the information-theoretical aesthetics. Foundation and application in the text theory)*. 1969

[8] Bentley, Peter ; Corne, David: *Creative evolutionary systems*. Morgan Kaufmann, 2002

[9] Biles, John: GenJam: A genetic algorithm for generating jazz solos. In: *Proceedings of the International Computer Music Conference* INTERNATIONAL COMPUTER MUSIC ACCOCIATION, 1994, S. 131–131

[10] Birkhoff, George D.: *Aesthetic measure*. Cambridge, Mass., 1933

[11] Bradski, G.: The OpenCV Library. In: *Dr. Dobb's Journal of Software Tools* (2000)

[12] Cheney, Nick ; MacCurdy, Robert ; Clune, Jeff ; Lipson, Hod: Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In: *Proceedings of the 15th annual conference on Genetic and evolutionary computation* ACM, 2013, S. 167–174

[13] Chervenski, Peter ; Ryan, Shane: *MultiNEAT*. `http://www.multineat.com/`

[14] Clune, Jeff ; Lipson, Hod: Evolving three-dimensional objects with a generative encoding inspired by developmental biology. In: *ACM SIGEVOlution* 5 (2011), Nr. 4, S. 2–12

[15] Colton, Simon ; Wiggins, Geraint A.: Computational creativity: the final frontier? In: *ECAI*, 2012, S. 21–26

[16] Cover, Thomas M. ; Thomas, Joy A.: *Elements of information theory*. John Wiley & Sons, 2012

[17] Deb, Kalyanmoy ; Pratap, Amrit ; Agarwal, Sameer ; Meyarivan, TAMT: A fast and elitist multiobjective genetic algorithm: NSGA-II. In: *Evolutionary Computation, IEEE Transactions on* 6 (2002), Nr. 2, S. 182–197

[18] Deng, Li ; Yu, Dong: Deep learning: methods and applications. In: *Foundations and Trends in Signal Processing* 7 (2014), Nr. 3–4, S. 197–387

[19] DiPaola, Steve ; Gabora, Liane: Incorporating characteristics of human creativity into an evolutionary art algorithm. In: *Genetic Programming and Evolvable Machines* 10 (2009), Nr. 2, S. 97–110

[20] Doncieux, Stephane ; Mouret, Jean-Baptiste: Beyond black-box optimization: a review of selective pressures for evolutionary robotics. In: *Evolutionary Intelligence* 7 (2014), Nr. 2, S. 71–93

[21] Dubbin, Greg A. ; Stanley, Kenneth O.: Learning to dance through interactive evolution. In: *Applications of Evolutionary Computation*. Springer, 2010, S. 331–340

[22] Eiben, Agoston E. ; Smith, James E.: *Introduction to evolutionary computing*. Springer Science & Business Media, 2003

[23] Elman, Jeffrey L.: *Incremental learning, or the importance of starting small*. University of California, San Diego, 1991

[24] Gauci, Jason ; Stanley, Kenneth O.: Autonomous evolution of topographic regularities in artificial neural networks. In: *Neural computation* 22 (2010), Nr. 7, S. 1860–1898

[25] Golberg, David E.: Genetic algorithms in search, optimization, and machine learning. In: *Addion wesley* 1989 (1989)

[26] Goldberg, David E.: Simple genetic algorithms and the minimal, deceptive problem. In: *Genetic algorithms and simulated annealing* 74 (1987), S. 88

[27] Gomez, Faustino ; Miikkulainen, Risto: Incremental evolution of complex general behavior. In: *Adaptive Behavior* 5 (1997), Nr. 3-4, S. 317–342

[28] Gomez, Faustino ; Schmidhuber, Jürgen ; Miikkulainen, Risto: Efficient non-linear control through neuroevolution. In: *Machine Learning: ECML 2006*. Springer, 2006, S. 654–662

[29] Gould, Stephen J. ; Vrba, Elisabeth S.: Exaptation-a missing term in the science of form. In: *Paleobiology* (1982), S. 4–15

[30] Greenfield, Gary: On the origins of the term computational aesthetics. In: *Proceedings of the First Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging* Eurographics Association, 2005, S. 9–12

[31] Hanson, N.R.: *Patterns of Discovery: An Inquiry into the Conceptual Foundations of Science*. Cambridge University Press, 1958. – ISBN 9780521092616

[32] Hastings, Erin ; Guha, Ratan ; Stanley, Kenneth O.: Neat particles: Design, representation, and animation of particle system effects. In: *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on* IEEE, 2007, S. 154–160

[33] Heijer, Eelco den: Evolving Art using Measures for Symmetry, Compositional Balance and Liveliness. In: *IJCCI*, 2012, S. 52–61

[34] Heijer, Eelco den ; Eiben, AE: Using aesthetic measures to evolve art. In: *Evolutionary Computation (CEC), 2010 IEEE Congress on* IEEE, 2010, S. 1–8

[35] Hoover, Amy K. ; Szerlip, Paul A. ; Norton, Marie E. ; Brindle, Trevor A. ; Merritt, Zachary ; Stanley, Kenneth O.: Generating a complete multipart musical composition from a single monophonic melody with functional scaffolding. In: *International Conference on Computational Creativity*, 2012, S. 111

[36] Husbands, Phil ; Jermy, Giles ; McIlhagga, Malcolm ; Ives, Robert: Two applications of genetic algorithms to component design. In: *Evolutionary computing*. Springer, 1996, S. 50–61

[37] Johanson, Brad ; Poli, Riccardo: *GP-music: An interactive genetic programming system for music generation with automated fitness raters*. Citeseer, 1998

[38] Johnson, Colin G.: Fitness in evolutionary art and music: what has been used and what could be used? In: *Evolutionary and Biologically Inspired Music, Sound, Art and Design*. Springer, 2012, S. 129–140

[39] Jolion, Jean-Michel: Images and Benford's law. In: *Journal of Mathematical Imaging and Vision* 14 (2001), Nr. 1, S. 73–81

[40] Koza, John R.: *Genetic programming: on the programming of computers by means of natural selection*. Bd. 1. 1992

[41] Krčah, Peter: Solving deceptive tasks in robot body-brain co-evolution by searching for behavioral novelty. In: *Advances in Robotics and Virtual Reality*. Springer, 2012, S. 167–186

[42] Krotkov, Eric: Focusing. In: *International Journal of Computer Vision* 1 (1988), Nr. 3, S. 223–237

[43] Lehman, Joel ; Stanley, Kenneth O.: Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In: *ALIFE*, 2008, S. 329–336

[44] Lehman, Joel ; Stanley, Kenneth O.: Revising the evolutionary computation abstraction: minimal criteria novelty search. In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation* ACM, 2010, S. 103–110

[45] Lehman, Joel ; Stanley, Kenneth O.: Abandoning objectives: Evolution through the search for novelty alone. In: *Evolutionary computation* 19 (2011), Nr. 2, S. 189–223

[46] Lehman, Joel ; Stanley, Kenneth O.: Beyond open-endedness: Quantifying impressiveness. In: *Artificial Life* Bd. 13, 2012, S. 75–82

[47] Li, Yang ; Hu, Chang-Jun: Aesthetic learning in an interactive evolutionary art system. In: *Applications of Evolutionary Computation*. Springer, 2010, S. 301–310

[48] Ligthart, Guido ; Groen, Frans C.: A comparison of different autofocus algorithms. In: *Proc. Sixth International Conference on Pattern Recognition*, 1982, S. 597–600

[49] Machado, Penousal ; Cardoso, Amílcar: Computing aesthetics. In: *Advances in Artificial Intelligence*. Springer, 1998, S. 219–228

[50] Machado, Penousal ; Cardoso, Amílcar: All the truth about NEvAr. In: *Applied Intelligence* 16 (2002), Nr. 2, S. 101–118

[51] Machado, Penousal ; Romero, Juan ; Cardoso, Amílcar ; Santos, Antonino: Partially interactive evolutionary artists. In: *New Generation Computing* 23 (2005), Nr. 2, S. 143–155

[52] Machado, Penousal ; Romero, Juan ; Santos, María Luisa ; Cardoso, Amílcar ; Manaris, Bill: Adaptive critics for evolutionary artists. In: *Applications of evolutionary computing*. Springer, 2004, S. 437–446

[53] Mahendran, Aravindh ; Vedaldi, Andrea: Understanding deep image representations by inverting them. (2014)

[54] Mántaras Badia, Ramon López de: Creatividad computacional. In: *Arbor* 189 (2013), Nr. 764, S. a082

[55] Martin, Worthy N. ; Lienig, Jens ; Cohoon, James P.: Island (migration) models: evolutionary algorithms based on punctuated equilibria. In: *Handbook of evolutionary computation* 6 (1997), Nr. 3

[56] Matkovic, Kresimir ; Neumann, László ; Neumann, Attila ; Psik, Thomas ; Purgathofer, Werner: Global Contrast Factor-a New Approach to Image Contrast. In: *Computational Aesthetics* 2005 (2005), S. 159–168

[57] McCormack, Jon: Open problems in evolutionary music and art. In: *Applications of Evolutionary Computing*. Springer, 2005, S. 428–436

[58] Moles, Abraham: Information theory and esthetic perception. Trans. JE Cohen. (1968)

[59] Mouret, J-B ; Doncieux, Stéphane: Encouraging behavioral diversity in evolutionary robotics: An empirical study. In: *Evolutionary computation* 20 (2012), Nr. 1, S. 91–133

[60] Mouret, Jean-Baptiste: Novelty-based multiobjectivization. In: *New horizons in evolutionary robotics*. Springer, 2011, S. 139–154

[61] Nelson, Gary L.: Sonomorphs: An application of genetic algorithms to the growth and development of musical organisms. In: *Proceedings of the Fourth Biennial Art & Technology Symposium* Bd. 155, 1993

[62] Neumann, L ; Sbert, M ; Gooch, B ; Purgathofer, W u. a.: Defining Computational Aesthetics. In: *Computational Aesthetics in Graphics, Visualization and Imaging* (2005), S. 13–18

[63] Nguyen, Anh ; Yosinski, Jason ; Clune, Jeff: Innovation engines: Automated creativity and improved stochastic optimization via deep learning. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, 2015

[64] Nishino, Hiroaki ; Takagi, Hideyuki ; Cho, Sung-Bae ; Utsumiya, Kouichi: A 3D modeling system for creative design. In: *Information Networking, 2001. Proceedings. 15th International Conference on* IEEE, 2001, S. 479–486

[65] Nolfi, Stefano ; Floreano, Dario: *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press, 2000

[66] Oudeyer, P-Y ; Kaplan, Frédéric ; Hafner, Verena V.: Intrinsic motivation systems for autonomous mental development. In: *Evolutionary Computation, IEEE Transactions on* 11 (2007), Nr. 2, S. 265–286

[67] Pedregosa, F. ; Varoquaux, G. ; Gramfort, A. ; Michel, V. ; Thirion, B. ; Grisel, O. ; Blondel, M. ; Prettenhofer, P. ; Weiss, R. ; Dubourg, V. ; Vanderplas, J. ; Passos, A. ; Cournapeau, D. ; Brucher, M. ; Perrot, M. ; Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830

[68] Pertuz, Said ; Puig, Domenec ; Garcia, Miguel A.: Analysis of focus measure operators for shape-from-focus. In: *Pattern Recognition* 46 (2013), Nr. 5, S. 1415–1432

[69] Ray, Tapabrata ; Liew, Kim M.: Society and civilization: An optimization algorithm based on the simulation of social behavior. In: *Evolutionary Computation, IEEE Transactions on* 7 (2003), Nr. 4, S. 386–396

[70] Rigau, Jaume ; Feixas, Miquel ; Sbert, Mateu: Informational aesthetics measures. In: *IEEE Computer Graphics and Applications* (2008), Nr. 2, S. 24–34

[71] Risi, Sebastian ; Cellucci, Daniel ; Lipson, Hod: Ribosomal robots: Evolved designs inspired by protein folding. In: *Proceedings of the 15th annual conference on Genetic and evolutionary computation* ACM, 2013, S. 263–270

[72] Risi, Sebastian ; Vanderbleek, Sandy D. ; Hughes, Charles E. ; Stanley, Kenneth O.: How novelty search escapes the deceptive trap of learning to learn. In: *Proceedings of the 11th Annual conference on Genetic and evolutionary computation* ACM, 2009, S. 153–160

[73] Roberts, Royston M.: Serendipity: Accidental discoveries in science. In: *Serendipity: Accidental Discoveries in Science, by Royston M. Roberts, pp. 288. ISBN 0-471-60203-5. Wiley-VCH, June 1989.* 1 (1989)

[74] Romero, Juan ; Machado, Penousal ; Santos, Antonino: On the socialization of evolutionary art. In: *Applications of Evolutionary Computing*. Springer, 2009, S. 557–566

[75] Romero, Juan ; Machado, Penousal ; Santos, Antonio ; Cardoso, Amilcar: On the development of critics in evolutionary computation artists. In: *Applications of Evolutionary Computing*. Springer, 2003, S. 559–569

[76] Romero, Juan J. ; Machado, Penousal: *The art of artificial evolution: a handbook on evolutionary art and music*. Springer Science & Business Media, 2007

[77] Rooke, Steven: Eons of genetically evolved algorithmic images. In: *Creative evolutionary systems* Morgan Kaufmann Publishers Inc., 2001, S. 339–365

[78] Ruiz-Mirazo, Kepa ; Umerez, Jon ; Moreno, Alvaro: Enabling conditions for 'open-ended evolution'. In: *Biology & Philosophy* 23 (2008), Nr. 1, S. 67–85

[79] Santos, Andrés ; Solorzano, C Ortiz d. ; Vaquero, Juan J. ; Pena, JM ; Malpica, Norberto ; Del Pozo, F: Evaluation of autofocus functions in molecular cytogenetic analysis. In: *Journal of microscopy* 188 (1997), Nr. 3, S. 264–272

[80] Saunders, Robert ; Gero, John S.: Artificial creativity: A synthetic approach to the study of creative behaviour. In: *Computational and Cognitive Models of Creative Design V, Key Centre of Design Computing and Cognition, University of Sydney, Sydney* (2001), S. 113–139

[81] Scha, Remko ; Bod, Rens: Computationele esthetica. In: *Informatie en Informatiebeleid* 11 (1993), Nr. 1, S. 54–63

[82] Schaul, Tom ; Bayer, Justin ; Wierstra, Daan ; Sun, Yi ; Felder, Martin ; Sehnke, Frank ; Rückstiess, Thomas ; Schmidhuber, Jürgen: PyBrain. In: *Journal of Machine Learning Research* 11 (2010), S. 743–746

[83] Schaul, Tom ; Sun, Yi ; Wierstra, Daan ; Gomez, Faustino ; Schmidhuber, Jürgen: Curiosity-driven optimization. In: *Evolutionary Computation (CEC), 2011 IEEE Congress on* IEEE, 2011, S. 1343–1349

[84] Schmidhuber, Jürgen: Low-complexity art. In: *Leonardo* (1997), S. 97–103

[85] Secretan, Jimmy ; Beato, Nicholas ; D Ambrosio, David B. ; Rodriguez, Adelein ; Campbell, Adam ; Stanley, Kenneth O.: Picbreeder: evolving pictures collaboratively online. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* ACM, 2008, S. 1759–1768

[86] Siegelmann, Hava T. ; Sontag, Eduardo D.: On the computational power of neural nets. In: *Journal of computer and system sciences* 50 (1995), Nr. 1, S. 132–150

[87] Sims, Karl: *Artificial evolution for computer graphics*. ACM, 1991

[88] Smith, Joshua R.: Designing Biomorphs with an Interactive Genetic Algorithm. In: *ICGA*, 1991, S. 535–538

[89] Stanley, Kenneth O.: Exploiting regularity without development. In: *Proceedings of the AAAI Fall Symposium on Developmental Systems* AAAI Press Menlo Park, CA, 2006, S. 37

[90] Stanley, Kenneth O.: Compositional pattern producing networks: A novel abstraction of development. In: *Genetic programming and evolvable machines* 8 (2007), Nr. 2, S. 131–162

[91] Stanley, Kenneth O. ; D'Ambrosio, David B. ; Gauci, Jason: A hypercube-based encoding for evolving large-scale neural networks. In: *Artificial life* 15 (2009), Nr. 2, S. 185–212

[92] Stanley, Kenneth O. ; Lehman, Joel: The Art of Breeding Art. In: *Why Greatness Cannot Be Planned*. Springer, 2015, S. 21–28

[93] Stanley, Kenneth O. ; Lehman, Joel: Why Greatness Cannot Be Planned. (2015)

[94] Stanley, Kenneth O. ; Miikkulainen, Risto: Evolving neural networks through augmenting topologies. In: *Evolutionary computation* 10 (2002), Nr. 2, S. 99–127

[95] Stanley, Kenneth O. ; Miikkulainen, Risto: Competitive coevolution through evolutionary complexification. In: *J. Artif. Intell. Res.(JAIR)* 21 (2004), S. 63–100

[96] Sun, Yu ; Duthaler, Stefan ; Nelson, Bradley J.: Autofocusing in computer microscopy: selecting the optimal focus algorithm. In: *Microscopy research and technique* 65 (2004), Nr. 3, S. 139–149

[97] Szegedy, Christian ; Liu, Wei ; Jia, Yangqing ; Sermanet, Pierre ; Reed, Scott ; Anguelov, Dragomir ; Erhan, Dumitru ; Vanhoucke, Vincent ; Rabinovich, Andrew: Going deeper with convolutions. (2014)

[98] Takagi, Hideyuki: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. In: *Proceedings of the IEEE* 89 (2001), Nr. 9, S. 1275–1296

[99] Thomsen, René ; Rickers, Peter ; Krink, Thiemo: A religion-based spatial model for evolutionary algorithms. In: *Parallel Problem Solving from Nature PPSN VI* Springer, 2000, S. 817–826

[100] Todd, Stephen ; Latham, William: Evolutionary art and computers. (1994)

[101] Unemi, Tatsuo: SBART 2.4: breeding 2D CG images and movies and creating a type of collage. In: *Knowledge-Based Intelligent Information Engineering Systems, 1999. Third International Conference* IEEE, 1999, S. 288–291

[102] Ursem, Rasmus K.: Multinational evolutionary algorithms. In: *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on* Bd. 3 IEEE, 1999

[103] Wolpert, David H. ; Macready, William G.: No free lunch theorems for optimization. In: *Evolutionary Computation, IEEE Transactions on* 1 (1997), Nr. 1, S. 67–82

[104] Woolley, Brian G. ; Stanley, Kenneth O.: On the deleterious effects of a priori objectives on evolution and representation. In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation* ACM, 2011, S. 957–964

[105] Woolley, Brian G. ; Stanley, Kenneth O.: A novel human-computer collaboration: combining novelty search with interactive evolution. In: *Proceedings of the 2014 conference on Genetic and evolutionary computation* ACM, 2014, S. 233–240

[106] Yosinski, Jason ; Clune, Jeff ; Nguyen, Anh ; Fuchs, Thomas ; Lipson, Hod: Understanding Neural Networks Through Deep Visualization. (2015)

[107] Zeiler, Matthew D. ; Fergus, Rob: Visualizing and understanding convolutional networks. In: *Computer Vision–ECCV 2014*. Springer, 2014, S. 818–833