DIPLOMARBEIT

Contaminated mutations: An analysis of controlled contaminations in design-oriented genetic algorithms in architectural design

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Diplom-Ingenieurs / Diplom-Ingenieurin
unter der Leitung

Oliver Schürer

E259
Institut für Architekturwissenschaften

Eingereicht an der Technischen Universität Wien
Fakultät für Architektur und Raumplanung
von

Christophe Libar
01009199

Wien, am eigenhändige Unterschrift

Abstract

Genetische Algorithmen (GAs) haben ihren Weg in die Welt des kreativen Designs gefunden. Ursprünglich vorrangig im Bereich des Ingenieurswesen angewendet, haben sie sich auch in kreativeren Disziplinen als wirksames Instrument für Diversifizierung und Innovation erwiesen. Im Kontext seines eigens konzipierten Imitation Games, bei dem ein Computer versucht, einem Menschen vorzutäuschen, ebenfalls ein Mensch zu sein, behauptete Alan Turing, dass der Computer aufgrund seiner extremen Genauigkeit entlarvt werden könnte. Daher schlug Turing vor, dass die Maschine absichtlich Fehler in das Gespräch einbauen sollte, um Fragestellende zu verwirren. Diese Arbeit beabsichtigt, eine ähnliche Logik auch auf GAs anzuwenden. Algorithmen, deren Optimierungsprozess nicht ausschließlich leistungsorientiert und funktionsbasiert ist, können einer kontrollierten Kontamination unterzogen werden, welche absichtlich Fehler in den Lösungsfindungsprozess des GA einführt, mit dem gleichen Ziel wie Turings Einwurf: die Einführung eines abstrakten, fehlerhaften Elements, das ein inhärenter Bestandteil der menschlichen Natur ist. Sogenannte kontrollierte Kontaminationen werden als neues Werkzeug verstanden, auf das Designer zurückgreifen können sollen. Der etwas paradoxe Vorschlag, bewusst Fehler in einen Optimierungsprozess einzuführen, wird zunächst durch Giorgio Agambens Sicht auf Kunst und Ästhetik untersucht, gefolgt von einer Analyse verschiedener wissenschaftlicher und kreativer Bereiche, die den Aspekt des Fehlerhaften in Bezug auf die menschliche Natur und Kultur untersuchen. Bei der anschließenden definierenden Beschreibung der kontrollierten Kontamination werden die Schlussfolgerungen der vorangegangenen Literaturstudie berücksichtigt. Abschließend wird die Funktionalität der vorgeschlagenen Kontaminationsfunktion anhand der Generation von Stadtkarten, die von Stan Allens Field Conditions inspiriert sind, in einem exemplarischen GA demonstriert, der die Wirksamkeit der vorgeschlagenen Kontamination empirisch untermauern soll.

Schlüsselwörter: genetischer Algorithmus, Kontamination, ästhetische Beurteilung, generatives Design

Abstract

Genetic algorithms (GAs) have found their way into the world of creative design. Originating in the domain of engineering, they have proven to be potent tools of diversification and innovation in more creative disciplines as well. In the context of describing his self-conceived Imitation Game, in which a computer attempts to deceive a human into thinking it is a human too, Alan Turing claimed that the computer could be exposed due to its radical accuracy. Hence, Turing proposed that the machine should embed deliberate mistakes into the conversation to confuse the interrogator. This paper proposes to apply the same logic to GAs. Algorithms whose optimisation process is not solely performance-driven, may be subjected to a controlled contamination, a function that introduces deliberate mistakes into the GA's solution finding procedure with the same goal as Turing's proposal: the introduction of an intangible, flawed element that is deeply inherent to human creation. The proposed controlled contaminations are understood as a tool that designers can revert to. The somewhat paradoxical suggestion of deliberate mistakes in an optimisation process is, in a first instance, examined through Giorgio Agamben's perspective on art and aesthetics, followed by an analysis of different scientific and creative fields that explore the aspect of the flawed in relation to human nature and culture. The subsequent practical delineation of the controlled contamination takes the conclusions of the preceding literature study into consideration. Finally, although limited in its complexity, an exemplary GA showcases the functionality of the incorporated contaminating function through the generation of maps that are inspired by Stan Allen's field conditions, with the aim of solidifying the claimed effectiveness of the proposed contamination empirically.

Keywords: genetic algorithm, contamination, aesthetic judgement, generative design

Acknowledgments

I have found a great deal of support in writing this master's thesis and I am sure that there are many people I should name personally, but the influences I have received that have helped me in the process are countless. Also, I like to keep things brief.

First of all, I would like to express my gratitude to my supervisor Senior Scientist Dipl.-Ing. Dr.techn. Oliver Schürer. His guidance and patience have been an invaluable source of motivation and inspiration for my endeavour. Our monthly talks have always shown me new and often daring directions.

Another thank you is directed towards the friends that I have met throughout the course of my studies, with whom I have shared many sleepless hours (Misery loves company, they say).

Finally, I want to thank my family, my love, and my dog in the sincerest way possible for the unconditional support over these last years and for always making time for me when I needed them.

Table of Contents

1 Introduction

Genetic Algorithms (GA) employ the basic principle of the survival of the fittest (Darwin, 1859) in Charles Darwin's evolution theory as their basis. They generate and compare different solutions to a problem with each other, mate the highest performing solutions, mutate their equivalent of a DNA and repeat the process until the generation's fitness is considered satisfactory. The aim is to find a solution that is as optimised as possible in respect to certain predefined criteria. Since their introduction in the 1960s by John H. Holland (Holland, 1992), GAs have been very effective in finding highly functional and often unusual and creative results to a myriad of tasks in a multitude of scientific and artistic fields. Their core effort lies in optimising an outcome by rooting out the flawed, the mistakes and errors through Darwin's principle.

Going back in time to the early 1950s, after one of the first supercomputers had recently shown its inherent might in deciphering complex code, expectations for the novel machine's future were soaring high. During those times, Alan M. Turing established an entire future field of computer science (Frankish & Ramsey, 2014) in his famous paper *Computing Machinery and Intelligence* (Turing, 1950) by asking whether machines would one day be able to think. Since the capacity of thought is difficult to verify empirically, Turing proposes a test setup that would have a computer and a human adversary seek to convince a human interrogator in a written conversation that they are human by answering the interrogator's questions. For the machine to succeed it would need impeccable linguistic and language-analytical skills as well as a reasonably broad knowledge base. Besides emphasising these primary competences, Turing addresses an additional concern that "the machine would be unmasked because of its deadly accuracy. The reply to this is simple. The machine (programmed for playing the game) would not attempt to give the right answers to the arithmetic problems. It would deliberately introduce mistakes in a manner calculated to confuse the interrogator" (Turing, 1950). Mistakes are thus described as a specifically human flaw that, if not occurring in a certain frequency or in relation to certain topics, would let humans effectively differentiate between humans and machines. Consequently, the virtue of making mistakes accomplishes to make machines and their output appear more human.

What can be deduced from this excerpt is of great meaning to the contemporary algorithmically aided design practice as it shifts the focus from a mere optimisation-driven process to an approach that embraces mistakes for the sake of a more flawed and human and a less pragmatic and controlled appearance. If one applies this line of thought to GAs, one might ask whether the field could draw profit from a certain amount of flawed input. Therefore, this paper proposes the introduction of a *contaminating function* that is set to counterbalance the calculation of a GA's fitness criterium towards a more nuanced, less optimisation-focused outcome with the aim of giving design solutions a difficult-to-measure but intrinsically human appearance. The proposed contaminations are meant to offset the notion of impurity that inevitably inhabits the concept of optimisation. This paper specifically focuses on GAs that are being applied in the architectural domain and other creative fields without the primary need for functionality. Other disciplines, whose sole focus is performance-driven, are not addressed because the procedure that is being proposed in this paper will not be applicable to them.

2 Background: current genetic algorithms

2.1 The basic principles of genetic algorithms

For the purpose of understanding this paper's way of reasoning, it is vital to appreciate the essential building blocks of GAs. As stated earlier, the underlying idea for GAs stems from evolutionary biology as defined by Darwin. Living species constantly fight for survival wherein only the fittest survive, i.e., the individuals that are best adapted to their environment through reproduction and mutation over many generations.

GAs abstract the aforementioned process into a varying number of stages: population initiation, crossover, mutation, fitness calculation and, finally, selection. Instead of determining which individual is the fittest to survive in a natural environment, GAs examine their generated phenotypes in regard to user-defined design goals e.g., an antenna that provides the highest connectivity, a floor plan with an ideal ventilation or a bridge whose stability is secured by the lowest possible amount of materials. These design goals are translated into quantitative parameters and represent the benchmark which the different solutions strive to achieve. Depending on the use case and the preferences of the programmer, a multitude of different algorithm layouts can be applied. For the purpose of simplicity, this paper chooses a basic prototype of a GA as described by *Genetic Algorithm Essentials* (Kramer, 2017).

The first stage, population initiation, randomly creates a varying number of solutions that ideally cover the entire solution space, meaning the set of solutions that do not exceed the boundaries of the design conditions. This first cluster of solutions is denominated as the initial population.

The following operator, crossover, takes pairs of two (or more) members of the population as parents, whose genetic material (e.g., in the form of single bits in string-focused algorithms) is then most often split up at a random point, shared with the other parent and recombined to form the parents' offspring. Numerous different methods for crossover are available to programmers: some crossovers select mates randomly, while others calculate the parents' fitness ahead of mating solutions with similar scores.

Before taking the offspring over to the next generation, their genetic material is mutated. The probability for a mutation to take place is calculated separately for each building block of the genetic material and is typically defined by floats between 0,001 and 0,1 (i.e., a probability of 0,1% to 10% for a building block to mutate). Mutation is an essential contributor to a GA's effectiveness as it ensures that additional, different building blocks are introduced into the randomly generated initial parent generation that were not present at the first stage. The algorithm's mutation rate happens to be one of its main regulating points: too many mutations might prevent the program from ever reaching its optimal goal, whereas a low rate might not bring enough diversity into the run and thus stale the evolution procedure.

In the next step, every population member's fitness is calculated in relation to the predetermined solution criteria with the intention of establishing an order that favours solutions with a high fitness score, thereby determining the fittest member of the population and sorting the remaining ones in descending order. Evidently, this process is crucial, and it is paramount that the fitness criterium is well-adjusted in view of the desired outcome. It should be noted that not only the proximity to the global optimal solution should be taken into account. Other criteria such as overall diversity and the vicinity to the optimum of a different fitness function in a multi-objective optimization should just as well be considered.

Finally, after all members have been evaluated, the ones with the highest fitness scores are chosen to produce a specific number of offspring which constitutes the next generation to be evaluated anew.

This loop of reproduction, mutation and selection continues as long until an optimal solution has been found that satisfies the designer's predetermined criteria.

Although the purpose of this brief description was a mere showcase of the basic principles of a GA, it should be noted that different modes of operation exist beyond the chosen example. As *An Introduction to Genetic Algorithms* (Mitchell, 1999) illustrates, the fitness calculation may very well be the first step after the population initiation, which renders the following crossover stage a more determined process that relies less on random probability as only the members with a similar fitness score mate with each other. The choice of the most potent method for a specific type of problem-solving gains in importance as the complexity of the solution finding increases. Consequently, algorithms that are bound to consider multiple optimisation parameters simultaneously depend on an efficient calculation method, which is strongly affected by the choice of the algorithm structure.

The following examples, which are meant to give an insight into a GA's practical implementation, were chosen in a manner that exhibits their functionality throughout different use cases. Every example is accompanied by a brief reflection on its relevance towards this paper's position.

2.2 Examples of genetic algorithms

2.2.1 The genetic algorithm in urban morphology

In their paper *Evolutionary algorithms for generating urban morphology: Variations and multiple objectives* (Makki, Showkatbakhsh, Tabony, & Weinstock, 2019), M. Makki et al. propose the use of a multi-objective evolutionary algorithm (a more advanced form of the basic GA) in the context of urban density increase on the example of the UNESCO world heritage site of Fes in Morocco. The algorithm generates a multitude of different approaches based on the researcher's predetermined requirements, which involve "(a) an increase in the area of neighbouring open spaces around each block cluster […], (b) an increase in the area of elevated public open spaces and walkways […], (c) an increase in the distance between upper and lower level open spaces […] and (d) a decrease in solar exposure on street level" (Makki, Showkatbakhsh, Tabony, & Weinstock, 2019) - some of the requirements standing in direct conflict with each other. The desired outcome is a set of different solutions that are expected to be equally explorative and exploitative; meaning that it is expected to contain diverse and optimised solutions. The resulting 25,000 designs prove to be highly adapted to the local circumstances albeit with partly polar emphases due to the conflicting nature of the requirements. Based on fitness criteria rankings, the best 10 solutions are analysed statistically and visualised for an in-depth inspection.

Moreover, specifically in the context of this paper, it should be mentioned "that although each generation produced 25 phenotypes, a small selection of these phenotypes were considered to be 'errors'; [...] these solutions were culled from the analyses and are not presented within this article's results" (Makki, Showkatbakhsh, Tabony, & Weinstock, 2019). Hence the authors made a deliberate choice to not include a number of non-computable results in their analysis although it might be argued that non-computable results may very well be results worthy of an analysis.

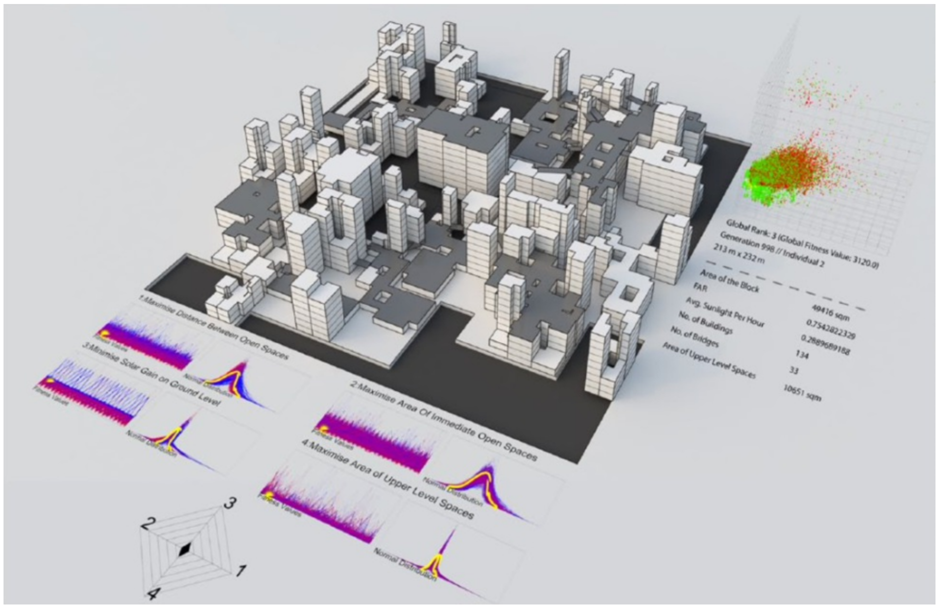

Global Rank: 3 (Global Fitness Value: 3120.0)
Generation 998 // Individual 2
213 m x 232 m
Area of the Block 49416 sqm
FAR 0.7542822329
Avg. Sunlight Per Hour 0.2889689188
No. of Buildings 134
No. of Bridges 33
Area of Upper Level Spaces 10651 sqm
1.Maximise Distance Between Open Spaces
2.Maximise Area Of Immediate Open Spaces
3.Maximise Solar Gain on Ground Level
4.Maximise Area of Upper Level Spaces
Normal Distribution
1
2
3
4

Figure 1: Visualised solutions with corresponding graphs (Makki, Showkatbakhsh, Tabony, & Weinstock, 2019)

2.2.2 <u>The genetic algorithm in automated floor plan generation</u>

M. Nisztuk and P. Myszkowski present a novel approach to automated floor plan generation (AFPG) in their article *Hybrid Evolutionary Algorithm applied to Automated Floor Plan Generation* (Nisztuk & Myszkowski, 2019). With the intent to aid architects in the early conceptual planning stages of a project by providing them with a variety of algorithmically generated functional and "non-obvious and surprising solutions" (Nisztuk & Myszkowski, 2019) for individual floor plans. Nisztuk and Myszkowski propose the integration of a hybrid genetic and greedy algorithm into existing CAAD software as well as the creation of a cloud-based web-application. The method in its current state takes only a single design criterion into consideration - the boundary and layout compactness. Nonetheless the algorithm does respect additional architectural qualities such as "topological connectivity, room side aspect ratios, and division into functional zones and the room areas to the total floor plan area ratio" (Nisztuk & Myszkowski, 2019). As expected by the authors, the application delivers reliable and computationally effective results, allowing designers to swiftly obtain an impression of feasible floor plans.

Again, it should be noted that the article emphasises that the computer based AFPG method is "a repetitive search for an optimal solution that meets the criteria and the design constraints" and that only a computer "is adapted to perform many repetitive operations without being susceptible to analytical errors" (Nisztuk & Myszkowski, 2019). A clear boundary is drawn between the optimising machine and the error committing human; the former being the preferred executor of certain tasks.

2.2.3 The genetic algorithm in poetry

In an attempt to show that GAs are applicable to a great variety of creative work, the publication *Using genetic algorithms to create meaningful poetic text* (Manurung, Ritchie, & Thompson, 2012) by R. Manurung, G. Ritchie and H. Thompson illustrates the GA's flexibility in light of a universal, yet often formalistic art form. While the preceding examples served as practical demonstrations for the application of GAs in architecture, the present algorithm shows that this type of machine-generated artistic expression is able to produce meaningful content, without the strict boundaries of building regulations, practicality or physics.

The authors establish specific prerequisites that the algorithm must adhere to, namely *grammaticality*, *meaningfulness* and *poeticness* – three categories that the program's outcome is evaluated on. They state that "A poem must be syntactically well-formed. […] A poem must intentionally convey some conceptual message that is meaningful under some interpretation. […] A poem must exhibit features that distinguishes it from non-poetic text" (Manurung, Ritchie, & Thompson, 2012). Through *lexicalised tree-adjoining grammar*, the program creates sentences that are subsequently assessed by the GA's fitness criteria. Although the presented method does not yet produce satisfying results, it achieves to accomplish a reasonable score in the aforementioned categories (e.g., "They play. An expense is a waist. A lion, he dwells in a dish. He dwells in a skin. A sensitive child, he dwells in a child with a fish." (Manurung, Ritchie, & Thompson, 2012)). However, further research is bound to yield improved results.

The article's setting is pertinent to the context of this paper as it illustrates a GA's attempt to optimise modes of speech with the aim of appearing natural to the observer. In this specific case, errors would provoke humans to question the validity of the results since classic poetry follows strict rules a human would always adhere to (except when trying to convey meaning through intentional mistakes – something a machine might try to simulate in an advanced version of the program through the *meaningfulness* variable).

2.2.4 The genetic algorithm in parametric design

Digital parametric design forms an excellent connection point for GAs as is exhibited in J. Harding and C. Brandt-Olsen's article *Biomorpher: Interactive evolution for parametric design* (Harding & Brandt-Olsen, 2018) as the authors demonstrate their Biomorpher-plugin for Grasshopper, which seeks to create an application for user-controlled evolutionary parametric design. Since it is argued that architecture seldom presents designers with "explicit objective functions" (Harding & Brandt-Olsen, 2018) – thereby making it difficult to define universal fitness criteria for a design approach using GAs – the authors propose the use of an interactive evolutionary algorithm. This type of GA requires the user to evaluate the phenotypes at each computed generation by manually choosing those that meet the designer's requirements the most. The possibility of a more standard performance-based selection through a performance evaluation node is nonetheless integrated, leaving the designer the choice of the required approach. The outcome is a functioning user-interface-based program that lets designers explore a variety of possible design solutions for floor plans, volume models etc. that may be further manually individualised without the GA's input through common parametric modelling.

The proposed approach for a more interactive integration of GAs into the design workflow focuses less on optimisation as it does on exploration. In bypassing objective performance criteria in favour of subjective preferences, the program opens itself up to the possibility of committing mistakes in relation to an optimal solution. In entrusting part of the responsibility for the choices that are being made to the user it could be argued that the approach embraces and incorporates the human and possibly flawed aspect of a more common, standard design process.

2.2.5 What can be deduced

The presented examples stemming from diverse fields of research and producing a diverse set of outputs showcase how GAs often touch the topic of errors in relation to human perception. Since this specific relation is not an imminent component of the program's solution-finding, it is not addressed in research papers, even though it evidently plays an underlying role in the thoughts that go into the algorithm's design. By their very nature, GAs need to assess their work continually with regard to a desired outcome. In doing so, the algorithms are a proxy to the designer's decision-making process – the designer being the authority that created the approximate boundaries of what is acceptable and what is to be discarded beforehand. The algorithms may take their liberties in how they interpret the space in between those boundaries but ultimately, they strictly adhere to the normativity established by their programmer. This circumstance is of course a strength of GAs, as they promote diversity in the solution space while striving to optimise within the prescribed design parameters.

Nonetheless, it is this author's strong belief that the relationship between the produced errors and the algorithms' discrimination method is in need of reformation. Going further, as the algorithms are merely a proxy to the designer's wishes, it is the designer that requires a shift in their emphasis as well. In the framework of an optimisation-driven process, the very concept of a flaw becomes an unacceptable occurrence, undesired and proscribed, although potentially only being recognisable through the machine's watchful eye. The flaw's natural occurrence in human designs is repressed.

2.3 Intermediately, a short note on the matter of architecture

It might be notable that one essential field is seldom addressed in the course of the preceding explanations, as they primarily focus on the workings of GAs, the breakdown of the terminology's various modes of comprehension and the research's current orientation. Albeit of its apparent absence, architecture always takes a central place in the discussion as two separate, split entities, namely aesthetics and functionality – both cornerstones of the architectural since Vitruvius – that encircle it thematically and pass on facts that are true for either of them to the architectural domain.

Instead of solely focusing on the architectural, this paper means to address art and design in general as confining itself to the former would entail a disregard for valuable insights that are true for art and architecture alike. It is certainly true that architecture places higher demands on an algorithm's generative functionality as, for instance, still life paintings. While the fine arts reflect primarily on aesthetics and meaning, a generated architectural design is required to consider functional, ecological, economical etc. elements in addition to its aesthetic appeal. Even so, discoveries in the aesthetic domain are certainly of value to architecture.

Consequently, architecture is seldom explicitly addressed in the following chapters. The reader is asked to keep in mind that observations on art and aesthetics should be applied in varying degrees to the architectural domain, specifically one that focuses on architecture as art; the same is true for the more functionality-focused findings in later chapters.

3 Methodology

The aim of this research is to learn whether Alan Turing's concept of deliberately introduced mistakes in a conversational environment is an effective approach to render the output of design-oriented genetic algorithms more approachable to its human users. As has been briefly explained in the introduction, Turing assumes that machines "would deliberately introduce mistakes in a manner calculated to confuse the interrogator" (Turing, 1950). This statement appears plausible in the specific context of Turing's Imitation Game, but less so in the framework of algorithmically optimised design. And yet, this paper seeks to establish a justification for its somewhat paradoxical hypothesis through the analysis of philosophical and social studies, as well as aesthetic and scientific fields that, in varying degrees and with different strategies, appreciate and value the idea of the flawed as a genuine benefit. In summary, it is this paper's aim to justify and establish the principle of a contaminating function that deliberately introduces mistakes into the output of a given design-oriented GA. The proposed contaminations are meant to alter the optimised aesthetic features of an algorithmically generated design by way of reinstating a number of flaws that have been discarded during the algorithm's optimisation process, with the purpose of rendering the algorithm's output more approachable.

Having begun with providing a broad overview of contemporary GAs in the preceding chapter with the aim of explaining the modus operandi of current GAs, this paper examined specific examples as a means to solidify the reader's understanding of the matter while at the same time highlighting some of the keypoints that put the showcased GAs in a relation with this paper's hypothesis.

In order to justify the aforementioned contaminations of algorithmic optimisation, it is crucial to examine the notion of aesthetics. Therefore, a profound analysis of Giorgio Agamben's *The Man Without Content* will set the criteria of how this paper frames the perceived understanding of aesthetic judgement and art in general with the intention of exposing a number of inconsistencies in the application of aesthetic norms in GAs. This approach aims to broaden the generative field's established conventions and to render them more amenable to novel concepts such as this paper's proposal. Thus, a special focus will be set on the definition of aesthetic judgement, (art) critics, the role of the artist and the spectator, a design's history and, finally, failure. A relationship between these concepts and GAs will be established, so that the rethinking of these terms will enable the rethinking of a GA's purpose.

The following part pivots its focus to selected scientific and aesthetic domains: Turing's foray into artificial intelligence, the social-anthropological view on the discernment of functioning technologies, the field of neuroaesthetics, the philosophy of chemistry and the aesthetic of wabi sabi in combination with mathematics. In examining these fields' respective principal dispositions through critical literature, a link to GAs will be drawn with each one to reveal the parallels of their particular approaches. While GAs are based on the fundamental principle of evolution (or rather the human perception thereof) it is standing to reason that, going forward, additional scientific disciplines may very well influence the conventional doctrines of GAs. It will be demonstrated how each of these disciplines can be an inspiration for a tool such as the proposed contaminating function.

Having drawn impulses from these other fields, this paper will delineate the nature of the proposed contaminating function. Basing itself on the insights gained through the preceding literary study, the rules of the implementation, the behaviour and the goals of such a function will be established. This is an explorative approach because of the experimental and paradoxical nature of the hypothesis.

Once the concept and the functionality of the contaminating function have been developed, it will be put to test in an empirical experiment. A program incorporating the newly implemented function will demonstrate its effects. Conceived as a rudimentary, abstract visual experience, the program is not conceived to deliver unambiguous and undisputable proof for the dominance of the new function. However, it is expected to showcase outcomes that may present several improved features compared to standard algorithms' solutions. It is certain that more elaborate versions of the contaminating function bear greater benefits in more advanced programs. Yet, in the scope of this work, it is not possible to attain a level of refinement that would be required for more elaborate algorithms.

The principal instrument of the present research is the comprehensive literature analysis that frames and deconstructs this work's primary concepts. The extensive study of Giorgio Agamben's writings serves as foundation upon which further topics can be discussed. These matters are explored with the help of scientific papers and books and deliver a comprehensive understanding of their respective research area that can be used to derive essential inspirations for the proposed contaminating function. Each domain can be seen as an argument for algorithmic contaminations in GAs. The second instrument of this paper, the empirical experiment in the form of a generative program, is written for the sole purpose of substantiating the features of the newly conceived function. Through the examination the program's visual output, the effectiveness of the contaminating function is assumed to be demonstrated.

4 Giorgio Agamben on Art, its Criticism and Aesthetics

Following the strict rules set by their programmer, genetic algorithms discriminate between what lies inside the boundaries of desirable design parameters and what lies beyond. In the process of creating a generated population's offspring, the algorithm judges each iteration's suitability and based on this judgement, either allows an iteration to procreate or discards it to be forgotten in the course of evolution. Design-oriented GAs are necessarily bound to make aesthetic judgements when considering an iteration's worthiness. While the desired design's outline is only vaguely determined by the human programmer, it is the algorithm that makes the ultimate decision whether an iteration meets the stipulated prerequisites – which is especially true for multi-objective algorithms, where different parameters are frequently in conflict with one another, leaving the definitive choice on how to proceed entirely up to the algorithm.

This established procedure is perfectly accepted throughout the field as it is reminiscent of how art in general is being looked upon these days. However, since this paper is questioning the purely optimisation-driven approach of current GAs, the rigid concept of the aforementioned procedure as such can be seen as a rather limited approach to how aesthetic judgement is set to occur. In his first publication, *L'uomo senza contenuto* (Agamben, 1999) (The Man without Content), Giorgio Agamben addresses the status of art in modern society. His thoughts he expresses often collide with the prevalent understanding of the commonly view on art, as well as its value and validation. What follows is a superposition of Agamben's assessments on aesthetic judgement in art and the principles of aesthetic judgement that are conducted by GAs with the aim of exhibiting more uncommon perspectives that might influence the customary techniques in conventional algorithms.

4.1 Aesthetic judgement and the critic

The art critic's main objective is to discern non-art from art and to present his evaluation of the latter to the public. The functioning of a GA then is not that distinct from an art critic's task. Its fitness evaluation process with the subsequent further analysis of strong population members, respectively the discarding of the weaker ones is analogous to the critic's metier. However, Agamben deconstructs the art critic's underlying principle and with it the very understanding of how art is actually defined as art: that is to say that art is essentially determined by what it is not, or what non-art is – undoubtedly a particularly dualistic way of thinking, but a perspective that is remarkably fitting to the binary functioning of a GA.

In stating that most of what is chronicled of the art critics' work from the nineteenth century is not about good artists, but rather about the ones that might be classified as unexceptional, Agamben addresses a paradoxical oddity. If one were to retrospectively base the value of those past artists on how much has been handed down about them, one would come to the conclusion that "Stendhal and Flaubert must be much inferior to Charles de Bernard, Vinet, Mole, Ramond, and other third-rate writers" (Agamben, 1999). The entire nineteenth century then seems to be influenced "by the principle that the good critic must go wrong on the good writer" (Agamben, 1999). And yet, while this practice seems rather grim at first, it is exactly what the art critic must do: bringing "art back to its shadow-if, by distinguishing art from non-art" (Agamben, 1999). Agamben suggests that if critics did not "make of the latter the content of the former and thus risk confusing them, our aesthetic idea of art would lose all consistency" (Agamben, 1999). Undoubtedly, this creates a rather intriguing and apparently paradoxical duality. The common understanding of art is only able to exist because of what art is not. Art and non-art are bonded, the one informs the other and with it our aesthetic judgement. Following this logic, a GA's discarded solutions form entities that define the optimised solutions. One might argue that the supposedly useless members of the population should consequently be assigned a higher value instead of merely being forgotten in the course of the algorithm's runtime. Given the interwoven nature of the evolutionary process's opposing outcomes, there is a significant probability that an evaluation of the discarded could hold valuable data that might be of use to the solutions that bear a higher fitness score. It is Agamben's firm belief that one thing always stands in relation to its diametric counterpart even if that thing actively tries to distance itself from its disparate self. He illustrates this ongoing relationship through the example of the nonverbal being unable to free itself from language. "In an analogous fashion, language also holds man in its ban insofar as man, as speaking being, has always already entered into language without noticing it. Everything that is presupposed for there to be language (in the forms of something nonlinguistic, something ineffable, etc.) is nothing other than a presupposition of language that is maintained as such in relation to language precisely insofar as it is excluded from language." (Agamben, 1995) Similarly, the discarded solutions remain in a *Beziehung* with those that are passed on in the evolutionary process, explicitly because they are being excluded from it. And yet, this intricate relationship is seldom explored in the common GA's setup, which focusses solely on the optimised outcome. Even though if does not explore the nature of this unbreakable bond it is the algorithm or the critic that enable it by defining the subject – be it art or algorithmically produced solutions – through its negation.

It is then only through the critic bringing art to its negation, to "this shadow and this death that art (our aesthetic idea of art) sustains itself and finds its reality" (Agamben, 1999). Agamben draws the comparison to Ivan Karamazov's inquisitor who condemns the returned Christ so as to uphold a Christian world, asserting that Christ has become superfluous, because it is in truth Satan and the fear of damnation that maintains order among the Christian people. Should art effectively be bonded to non-art, or the thing that is to what it is not, and if art is indeed only able to sustain itself in its own shadow, it is clear that the common understanding of aesthetic judgement is an inadequate one as it fails to recognize the importance of the less visible, lower esteemed side. The question arises how GAs should be altered so that they may possess the capability to implement the consequences of such a cardinal redefinition of values. An answer might be found through Agamben's final assessment of art's reality and our understanding of it.

"Our appreciation of art begins necessarily with the forgetting of art", he states, because the tool of aesthetic judgement is needed to know a piece while it just as well prevents us to "penetrate its reality" and even worse, it leads us to the point of recognizing that "art's reality" is "pure and simple nothingness" (Agamben, 1999). Agamben concludes that art has come to solely being observed from its "dark side" and "that aesthetic judgment is then nothing other than the logos, the reunion of art and its shadow" (Agamben, 1999). Exerting aesthetic judgement as the mere "logos, the reunion of art and its shadow" (Agamben, 1999) is thus the critic's main aspiration. Commonly thought of as the entity that segregates the artful from the philistine it is indeed the exact opposite. Art as such and on its own is comprised of "pure and simple nothingness" (Agamben, 1999) and only in reconciling it with what art is not, it becomes identifiable as art. This is precisely the purpose of aesthetic judgment.

However, as stated before, GAs do not tend to what is deemed unworthy by their algorithmic critic. Their discarded solutions, essentially non-art, are forgotten. A reconciliation between the duality never takes place. Following Agamben's logic, this neglect should reduce the intelligibility behind the desired optimal outcome to nothingness. The aim of this paper is precisely to reunite art with non-art, the optimal with the discarded and thereby to guide GAs toward a critic's intrinsic purpose.

To continue on this path, it is nonetheless important to note a special circumstance of GAs that the common critic does not share. While critics usually tend to the works of other artists, GAs analyse their own creations. They unite the entities of the critic and the artist into a single actuality.

4.2 The unification of artist and spectator

According to Agamben, there is a stark discord between the artist's perspective and the perspective of the spectator. This kind of dissonance seems to be so prevalent that it can make an artist doubt the very nature of its work, should they dare to switch to the other side. This is illustrated by the example of the artist Frenhofer in Balzac's *The Unknown Masterpiece*, who, after letting two fellow artists get a glimpse of his frenetically elevated work of art, is subjected to the point of view of the spectator, thereby making the "integrity of his work dissolve" (Agamben, 1999). Agamben describes the alternating transformation as follows: "Frenhofer becomes double. He moves from the point of view of the artist to that of the spectator, from the interested promesse de bonheur to disinterested aesthetics. [...] For it is not only Frenhofer that becomes double, but his work as well; [...] The side that faces the artist is the living reality in which he reads his promise of happiness; but the other side, which faces the spectator, is an assemblage of lifeless elements that can only mirror itself in the aesthetic judgment's reflection of it." (Agamben, 1999) As has been established before, GAs are not only creators but also the critics of their own work, meaning that they are necessarily bound to take on the role of the external spectator. This circumstance continuously confronts them with Frenhofer's fatal disposition. The art they create is subsequently split into two entities – one that promises its creator a perspective, a goal or simply happiness and one that seeks to satisfy the spectator's mere longing for beauty, a "disinterested beauty" (Agamben, 1999). One must understand, that according to Agamben, when assessing a piece of art, it should be one's goal to "to purify the concept of 'beauty' by filtering out the αϊσθησις, the sensory involvement of the spectator, and thus to consider art from the point of view of its creator" (The Man Without Content, 1999). However, in recent times, art has predominantly been examined through the eyes of the spectator instead of those of the artist and it is the spectator's yearning for the disinterested beauty that paves the way for the emergence of the now widely established phenomenon of taste. A "man of good taste" is consequently a person that can distinguish between "good and not so good art" (Agamben, 1999), basing their assessment on how well the piece of art that presents itself to them managed to satisfy their desire for beauty. It can be assumed that the logic behind GAs was developed in the same order, but in favour of computational simplicity (or even feasibility) one cannot expect that an algorithm designs with a *promesse de bonheur* in mind. Instead, it efficiently distinguishes between humbler and in some cases even measurable parameters. Although the algorithm is unmistakably a creator, ironically, it has no recourse on an actual creator's motivation – it is caught in an insurmountable split.

It should be specified that the algorithm being described as a creator does not grant it the same kind of agency as one would assign to a human. It is seen as a sophisticated tool that is certainly more autonomous and constructive than a painter's brush, but ultimately it carries out its programmer's bidding. Creation then is not to be understood as a mystic, divine deed, but should be read as a pragmatic interpretation of the ancient Greeks' understanding of poiesis, which was "the experience of production into presence, the fact that something passed from nonbeing to being, from concealment into the full light of the work" (Agamben, 1999). The algorithm produces something that has already existed as an idea (e.g., in its programmer's mind), which it merely brings from nonbeing to being.

However, in its creative process, the algorithm may observe its creations only through the spectator's perspective and with that, through the lens of taste. And since there are men of good taste, it is clear that men of bad taste must exist as well. Although shunned in the eyes of those that claim to possess good taste, men of bad taste are in a way redeemed if not praised by Agamben after citing the example of Jourdain in Molière's *Le bourgeois gentilhomme*. He suggests that art might indeed favour the "undifferentiated mold of bad taste" over "the precious crystal of good taste [...] as if art, entering the perfect receptive mechanism of good taste, lost that vitality that a less perfect but more interested mechanism is on the contrary able to preserve" (Agamben, 1999). The absence of good taste might thus enable the spectator to look beyond the concept of beauty and to free them from a perceived compulsion to expecting certain elements of beauty, thereby favouring a more interested perspective on the work of art.

Not only do GAs forcibly omit the artist's view on their quest for the promesse de bonheur in their assessment of a design, but they are also uninterested in the artwork itself by relentlessly trying to "perceive the point de perfection" (Agamben, 1999) as do all men of good taste. It could then be argued that the uncompromising quest for optimisation leads GAs astray and that they should change their modus operandi in favour of a less entitled evaluation of their work – one might even call it a laisser-faire approach. While it has been established that the GAs' harsh differentiation between art and non-art and the subsequent neglect of the latter deprive their creation of meaning, it becomes clearer that their universal way of discriminating between the tasteful and the tasteless as such lets them neglect their initial purpose. Thus, it is only by undermining their present principles that GAs may increase their potential.

4.3 Between the Angel of History and the Angel of Art

In the final chapter of his book, Agamben draws a link between art and history. He employs the image of two angels that represent art and history and bases their appearance and the interpretation of their vocation on the Angelus Novus painted by Paul Klee and analysed by Walter Benjamin and an angel from an engraving by Albrecht Dürer – the former is identified as the angel of history, the latter as the angel of art.

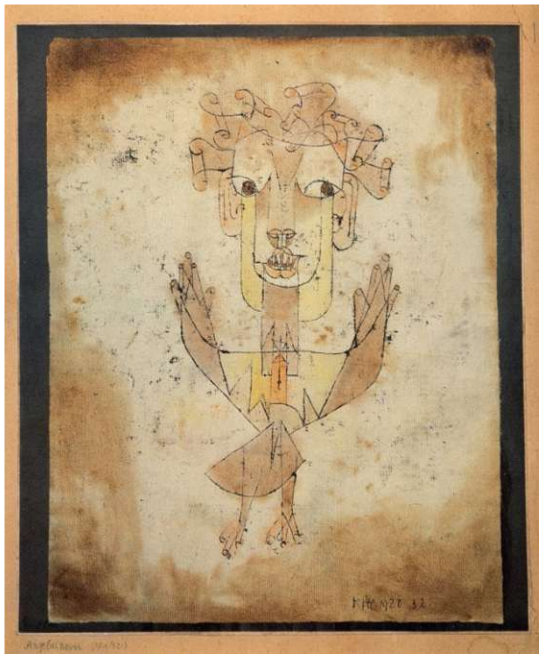


Figure 2: Paul Klee – Angelus Novus (1920).

According to Walter Benjamin, the angel of history's "*face is turned toward the past*", where "he sees one single catastrophe which keeps piling wreckage upon wreckage and hurls it in front of his feet"(Benjamin, 1996). He is unable to dwell and make sense of the destruction he witnesses, because "a storm is blowing from Paradise" that "irresistibly propels him into the future to which his back is turned" (Benjamin, 1996).

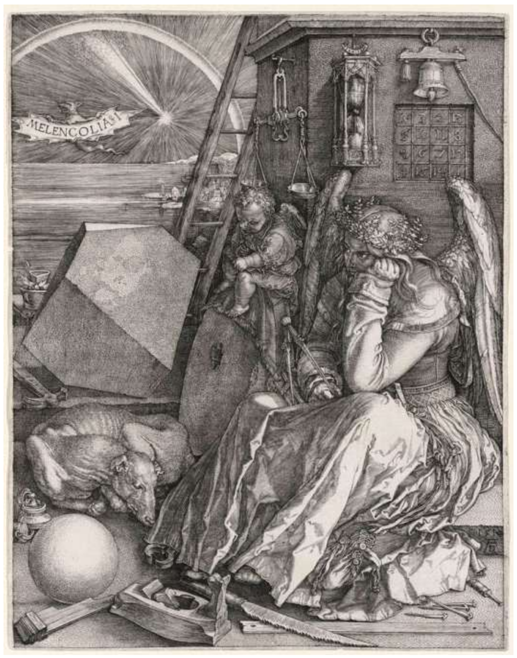


Figure 3: Albrecht Dürer – Melencolia (1514)

Dürer's angel, in turn, is sitting calmly and "looking ahead with an absorbed expression" (Agamben, 1999). Tools and everyday objects alike surround the angel, as the storm that plagues the angel of history, the storm of progress, has disappeared; he finds himself in "an atemporal dimension, as though something, interrupting the continuum of history, had frozen the surrounding reality in a kind of messianic arrest" (Agamben, 1999). In Agamben's allegory, both angels have lost agency over their respective remit: "the events of the past appear to the angel of history as a pile of indecipherable ruins" and "the utensils of active life and the other objects scattered around the melancholy angel have lost the significance that their daily usefulness endowed them with" (Agamben, 1999). Alienated, they have become "the cipher for something endlessly elusive" (Agamben, 1999). Evidently, history and art are placed within a relationship of dependency, albeit a relationship that never completely satisfies either one of the angels.

What is true for the general spectrum of art can be applied to the principle of a GA. In a similar fashion to the previous chapters, the angel allegory hints at the importance of the discarded past generations to the algorithm's optimal solution. While the allegory does not specifically allude to the nature of past generations, it appears rather evident that the aspect of being deemed as flawed represents an essential part of the past (this property being the primary reason the discarded members are part of the past). According to Agamben, the image of the past that the angel of art is able to reconstruct "is the alienated image in which the past finds its truth again only on condition of negating it, and knowledge of the new is possible only in the nontruth of the old" (Agamben, 1999). This suggests that the angels' image of the other one's metier is not entirely accurate. The reason behind this alienation is the angel of art's limitation of only being able to make the past "appear outside its real context on the day of aesthetic Last Judgment" which can only entail "its death (or rather, its inability to die) in the museum of aesthetics" (Agamben, 1999). This inescapable circumstance can only end in melancholy on the angel of art's side as "he has adopted alienation as his world; it is the nostalgia of a reality that he can possess only by making it unreal" (Agamben, 1999). Although this appears to be a troublesome fortune for the angel of art, it is a necessary and good one for GAs. An exact representation of its history would lead the optimal solution to be less than ideal in respect to its purpose, namely, to offer a certain degree of refinement. It needs to be seen which aspects of the past are to be passed on to the optimal solution.

The link between the two angels is conveyed through aesthetics. Replacing tradition, "it resolves the conflict between old and new" (Agamben, 1999), an equilibrium that is of utmost importance to mankind. "By destroying the transmissibility of the past, aesthetics recuperates it negatively and makes intransmissibility a value in itself in the image of aesthetic beauty, in this way opening for man a space between past and future in which he can found his action and his knowledge" (Agamben, 1999). Hence, mankind benefits from a degree of intransmissibility just as GAs do. "A space […] in which he can found his action and his knowledge" (Agamben, 1999) is the topos in which the GA's optimisation through innovation takes place. Nevertheless, as has been established, the angel of art never completely loses hold of the past and the accompanying nostalgia. A characteristic, that should – at least to some extent – be implemented into the algorithm.

4.4 The anthropomorphisation of failure

The final point is one that Agamben borrows from Nietzsche's *The Gay Science*, only to expand on it further. The argument appears to be nihilistic in its assertion as it posits that the world's ultimate state is chaos in that it presents us with "a lack of order, arrangement, form, beauty, wisdom, and whatever other names there are for our aesthetic anthropomorphisms" (Nietzsche, 2008). While the argument certainly pertains to the realm of aesthetics, it applies just as well to the more general strive of humankind to consolidate its faculties. Nietzsche states that all the aesthetic purpose humans may see in the world are mere anthropomorphisms that are by no means given. It is a realisation that bears universal doubt and entails an absence of meaning that churns humanity's convention of how it has tried to evaluate its world.

However, it is possible to reverse such a seemingly distressing realisation in itself, so that the ultimate outcome is not an absence of meaning but much more a gain thereof. Following Nietzsche's logic, even the irrefutable laws of nature lose their standing as laws since "there is nobody who commands, nobody who obeys, nobody who trespasses" (Nietzsche, 2008). The only guide of the animate and the inanimate through the universe is necessity, albeit without purpose. Considering this lack of purpose, it becomes clear that everything that occurs does not do so because it tries to follow a set of rules but rather because the surrounding influences make it necessary that the occurrence plays out in a specific way: there are no accidents, only consequences. Failure subsequently becomes a human creation, an anthropomorphisation of certain manifestations which exist inherently without bias. And yet, failures remain the rule in human endeavours while successful attempts are regarded as the exception. "When may we begin to 'naturalize' humanity in terms of a pure, newly discovered, newly redeemed nature?" (Nietzsche, 2008) is the question that Nietzsche raises thereupon, seeing that the reason of some developments' discarding is solely based on human arbitrariness and in no way in accordance with what is given by nature. The "newly discovered, newly redeemed nature" (Nietzsche, 2008) in question is one that would leave behind the entrenched and mystified notions of the world's functioning. This is where the devaluation of all meaning might indeed result in a gain thereof in the sense that the supposedly evident notions are no longer the limits to human thought and that there lies meaning beyond.

In the (more practical) scope of this paper it should be refrained from scrutinising the laws of nature, but it is evident that the technique of the GA might hold a certain degree of arbitrariness. It is however just as clear, that the results of GAs need to respect the required boundaries – GAs would lose their validity if that requirement was not given, as a simpler random generative algorithm without any method of elimination would subsequently suffice. Still, the aspect of Nietzsche's explanation that calls into question the current order of human thought enables the theoretical criticism of the mere existence of an unsuccessful endeavour. In consequence, it is reasonable to think of novel techniques that GAs might fulfil their tasks in a less restrictive manner. As was stated earlier: what lies beyond the current boundaries might be of gain to the conventional process.

4.5 Implications for GAs

The aforementioned concepts that touch a wide range of topics and disciplines in the creative and cultural sphere are being evoked for the simple reason of opening up the field and putting into question some tenets that are often taken for granted because of their perceived unshakable universality. GAs and much of the creative process follow the same patterns and have done so for quite some time. It is not intended to scrutinise this convention, as it surely and demonstratively is a fruitful one that has borne an immense variety of artworks throughout the ages. Nevertheless, it is Agamben's effort to question the most basic und internalized procedures, that should be seen as a point of departure to apply this thought to practical, applied techniques. To probe into this principle in relation to GAs makes sense in that it renders it possible to question the principal goal of a GA, namely, to optimise. As has been illustrated, optimisation is a purely normative concept that becomes only evident through human evaluation. Observed in an absolute vacuum, an algorithm's process of purification is then nothing more than a surrounding influence that pushes a design to evolve in a certain way.

Diverging from this pattern does not have to be labelled an accident as the divergence as such is an entirely neutral consequence of a set of specific actions.

Combining this idea with the line of thought that examines the relationship between an algorithm and its creation opens a wide spectrum for novel techniques of how programmers should tackle the conception of their algorithm. It is specifically this spectrum that is seen as an opportunity for this paper to propose its concept of *controlled contaminations*. In the traditional optimisation-focused approach, any deviation from the local optimum is eagerly discarded, whereas the newly introduced contaminations rather paradoxically aim to revalorize what has previously been discarded.

5 Of Errors and Impurity

While it has been established that the thought construct underlying GAs might profit from a set of new influxes, it is still unclear why *controlled contaminations* should take on the role of the innovator. After all, the introduction of faults into the optimisation process of a GA appears to be a counterintuitive if not a paradoxical measure. This chapter will try to justify the chosen tool that is set to explore this newly opened spectrum in the conception of GAs while challenging the field's general disposition towards flaws and impurity even further.

The present research into controlled contaminations draws its inspiration from Alan Turing's work, who, in a short comment that, to this date, has not been granted much attention, rightly ascertained that humans were able to differentiate between the humanly and the algorithmically created through the sheer accuracy and flawlessness of the algorithm's output. Turing applied and limited his assessment to a conversation between a human and a machine, whereas this paper argues that it is indeed pertinent to a variety of situations in a world that relies more and more on computer-generated works. Of course, Turing's original concern of humans not being able to differentiate between a human and an artificial discussion partner is not a primary consideration of this paper, but it introduces the question whether algorithmically generated output might lack a intangible aspect that inherently inhabits human design. It is through the exploration of Alan Turing's reflections and various other fields, which already incorporate in one variation or another the concept of a desirable flaw or controlled contaminations, that the following subchapters will try to justify the introduction of a similar system into a GA's mode of operation.

5.1 Mistakes, explained

Through the course of this work, a specific set of words, that address the subject of mistakes, has been frequently repeated. For the sake of variety, the word 'mistake' has been substituted with 'error', 'shortcoming', 'fault' and so forth. Although stemming from the same general locale, one would be right in saying that these words all bear a different meaning. Indeed, there are intricacies to every one of them that renders one more suitable for the context in which it is employed than another one. And yet, whenever they are used in the scope of this work, they try to evoke a very specific concept of mistake that will be explained in this brief chapter.

The overall most fitting term would certainly be 'flaw', which describes the attribute held by or attributed to a technology, a thought, a piece of art etc. that inhibits the bearer's performance aesthetically or functionally. Alternatives of the word 'flaw' are certainly not outright synonyms, but they do just as well carry the connotation of a diminished performance. While 'flaw' does not specify how the drop in performance came to be, 'mistake' or 'error' hint at a malfunction in a given process, which, in some cases, is a useful addition to the original meaning. Since there is a great variety of flaws with vastly differing magnitudes and implications it should be made clear what sort of flaw ought to be introduced into a GA's method of operation.

By definition, a GA will attempt to remove as many flaws as possible from its generated population during its run with the purpose of optimising the outcome. This basic principle certainly remains desirable as it drives the generated solutions towards the prerequisites given by the programmer. Naturally, many solutions with grave deficiencies are deleted from the design (for instance windows that serve as doors or antennas without any conductors) during the process. The chosen designation for this kind of flaws is *condition-breaking*, meaning that such a flaw would hinder a design to fulfil its function or to meet the given aesthetic prerequisites if the algorithm did not remove it. Whenever this paper calls for the preservation of mistakes, errors and the like, condition-breaking flaws are not the kind it sets out to maintain.

A valuable inspiration comes from Jenn Neilson's article *Can Moral Flaws Count as Aesthetic Virtues* which employs the terminology of the "surface moral flaw" (Neilson, 2012). The paper examines the influence of moral flaws in a work of art on the spectator's engagement. According to Neilson, the surface moral flaw consists of a discrepancy between the imaginary moral actuality a work of art exhibits and the audience's moral beliefs. Said discrepancy might cause the audience to disengage with the work of art. Far from stating that a surface moral flaw and the potential consequence of disengagement are necessarily an impediment to the work's effect, Neilson argues that, contrarily, if utilised in the right way, it may prompt a stronger engagement due to the more active reflection on the presented matter for the audience's part.

Given the thematic vicinity of Neilson's surface moral flaw to this paper's account in favour of the flawed in an otherwise optimised environment, it appears evident to incorporate an adapted version of Neilson's proposal. The *surface algorithmic flaw* then delineates small, non-functionality-breaking imperfections that might equip the algorithm's output with a given beneficial feature and at the very least entails a stronger engagement by the output's observer through the flaw's out of the ordinary impact.

In this paper, every argument for deliberate faults, mistakes, errors etc. in a GA's solution is an argument in favour of a surface algorithmic flaw, as defined in this chapter.

5.2 The Turing machine and the virtue of erring

Turing's influential paper on *Computing Machine and Intelligence* has been mentioned several times throughout this paper. One reason for the repeated mention being the fact that the initial idea for this paper's approach stems from a casual reading of the famous mathematician's work, while the second one lies in the immense importance of the paper and the conceptual changes it has led to up until this day. Alan Turing has often been described as the "Father of Artificial Intelligence" (Luger & Chakrabarti, 2017) whose ideas on thinking and learning machines have influenced generations of computer scientists. Turing considered it difficult to define the ability to think in the context of machines and proposed his *Imitation Game* instead, that would check whether a human would succeed in discerning the machine from a human in a written discussion. The capacity of thought as such, whose exact nature is discussed in philosophical and even theological terms should not be of interest in regard to machines, Turing argued. He considered it a much more effective approach to verify whether machines would ultimately accomplish to imitate humans in a way that would be indistinguishable to other humans, as this would imply that thought was indeed simulatable, since nobody would deny a human's capability to think.

A simulated thought is arguably something entirely distinctive of a thought whose originator had a consciousness of its own thinking. This is an argument that has long been made in relation to the question of whether machines could think, and it is an argument that Turing simply rejects as solipsistic. By contrast, philosopher Daniel Dennett refutes it on a more systematic level by suggesting that "if all the phenomena of human consciousness are explicable as 'just' the activities of a virtual machine realized in the astronomically adjustable connections of a human brain, then, in principle, a suitably 'programmed' robot, with a silicon-based computer brain, would be conscious, would have a self" (Dennett, 1991). The statement is naturally just as applicable to artificial intelligence programs without a robotic body. He expands on his thought by explaining that it is indeed hard to imagine a myriad of electromechanical connections producing some form of consciousness, but that it is surely just as hard to expect a network of electrochemical connections doing it – and yet that is precisely what occurs in the human brain.

In his endeavour, Turing addresses the common assertion that machines are incapable of mistakes (while being "tempted to retort: 'Are they any worse for that?'" (Turing, 1950)). However, mistakes are an integral part of the human condition and are bound to happen under certain circumstances, which would make it easy for the interrogator to expose the machine player, e.g., by asking it to solve an extremely challenging mathematical equation. In order to counteract this limitation, Turing proposes that the machine should purposely introduce a number of mistakes into its answers. It becomes clear that errors, flaws and mistakes are an essential part of human interaction in this context. Since the problem is hereby solved, Turing does not go into further detail in relation to his purposefully erring machine. Nonetheless, one should consider to investigate the issue at hand a little further by expanding on the process of erring, especially in relation to GAs.

The creation of possibly flawed solutions to a design task is a decisive facet of every GA. Based upon these less-than-optimal results, more optimised ones are being created that approach the predetermined design goal incrementally. There is no progress without erring. Yet, one might argue that every flawed solution is an optimum in itself much more than a mistake, as it is in the majority of cases more optimised than the members of the previous generations. Every iteration is then closer to the optimum than its parents, grandparents etc. After all, the final result of a GA might still display a number of flawed aspects. The algorithm might only have terminated because a particular maximum number of generations has been reached or the solution parameter's divergence is lying within a margin set by the programmer that seems adequate for the predetermined requirements (e.g., an algorithmically designed apartment building features direct sunlight for 60% of its units – while that number might just as well be 65%). This concern becomes even clearer if an algorithm needs to discriminate between opposing design goals (e.g., direct sunlight vs. natural cooling of a space). One or several parameters cannot reach their potential optimum due to the conflict with another parameter. They might very well be described as flawed, because of their inability to reach the design goal. In any case, it remains that mistakes are an integral part to GAs.

Alan Turing himself introduced the idea of learning algorithms: "It may be used to help in making up its own programmes, or to predict the effect of alterations in its own structure. By observing the results of its own behaviour it can modify its own programmes so as to achieve some purpose more effectively" (Turing, 1950). Through the observation of its own conduct, the machine should learn to adjust itself, so as to optimise future outputs. Again, one might ask whether an optimization requires actual errors or may just as well take place by improving an already functioning principle. "One must experiment with teaching one such machine and see how well it learns.", Turing explains the matter, and continues that "we normally associate punishments and rewards with the teaching process. [...] The machine has to be so constructed that events which shortly preceded the occurrence of a punishment signal are unlikely to be repeated, whereas a reward signal increased the probability of repetition of the events which led up to it" (Turing, 1950). Naturally, these punishment signals would occur after the child programme had committed a mistake – which suggests that Turing expected actual errors and hardly a smooth, gradual improvement that is based on already valid answers. It would not be an exaggeration if one would argue that the preceding description matches that of a very rudimentary deep learning algorithm with its adjusting weight layers. It is precisely this model that Turing considers best suited for his Imitation Game. Another aspect of Turing's concept is mutation: "It is probably wise to include a random element in a learning machine. A random element is rather useful when we are searching for a solution of some problem" (Turing, 1950). The importance of a GA mutating its solutions during each generation has already been discussed as it is an essential feature that prevents a narrow solution space.

The question might arise why these comparisons between Turing's initial concept and modern-day genetic and deep learning algorithms are being drawn. The reason is simple: it has been established that many of the aspects that Turing considered efficient have been ported directly to modern learning algorithms, often in a very literal way. This was of course a valid decision, seen that they have become immensely powerful problem-solvers, chiefly through learning from past mistakes or unfit solutions. However, – and this is the point of the matter – there is one facet of Turing's argument that has been largely overlooked, as there are two types of mistakes that the computer scientist refers to: those that take place in the learning process (equivalent to today's deep learning algorithm's training, respectively a GA's selection process) and those that are executed deliberately during the program's run during the Imitation Game. The latter is prominently omitted in modern algorithms. Alan Turing did not elaborate on how he intended his machine to execute the deliberate mistakes during the game, but one might imagine that it would have recourse to the mistakes committed during the learning period. Additionally, the machine would have to comprise a sort of memory that may access the answers it was originally taught to avoid – a process that has indeed been proposed by Turing in the section that outlines the punishment mechanism. To recall, the goal behind the machine's deliberate mistakes is to give it a more human appearance. It is this paper's intent to apply the same logic to today's learning algorithms, which, in their conception and build logic, are evidently very similar to Turing's machine, but neglect the discussed property entirely. It is clear that their outcome would be of a different kind than a natural conversation, but it would surely engender a rather intricate, quintessentially human quality.

Beyond the matter of mistakes in the Imitation Game, one might consider revaluating Turing's proposed learning concept. The most plausible implementation of machine learning at the time was the application of the prevalent teacher-centred teaching, meaning a strictly hierarchical dissemination of knowledge that is enforced through rewards and punishments by an authoritarian instructor. Given that this manner of teaching has become deprecated – specifically punishments are no longer considered a pedagogic tool (Geiger, 2000) – it would be interesting to explore the possibilities of more modern approaches to teaching and societal norms might hold for today's algorithms, which are still largely (yet loosely) based on Turing's initial concept. There are most certainly a number of strategies that try to depart from the established standards – e.g., unsupervised learning – and yet, one might wonder whether these welcome deviations could not be more substantial. The implementation of more unconventional algorithmic learning methods lies far beyond the scope of this paper but might be explored in future works.

5.3 Technology's most human facet

The Turing machine strives to simulate and imitate human interaction. In doing so, the technology is persistently being judged by the humans it was programmed to outplay. It succeeds once its counterplayer can no longer distinguish the machine from an actual human. Meanwhile, the role and reception of technology as a whole throughout societies has been the focus of numerous anthropologists. How technologies come to be and what is considered a successful technology have long been the cause for discussion. This section will examine anthropology's perspective on technology and the bias that is often in conjunction with technological development.

The anthropological approach proceeds on the assumption that technology is more than the plain application of a process seeking to attain a certain goal. Typically, a technology is defined as "any tool or technique, any physical equipment or method of doing or making, by which human capability is extended" (Schön, 1967). This characterisation includes (by today's standards) simple devices like a hammer, techniques such as traditional Japanese pottery or contemporary design algorithms that GAs are representatives of. Notably, this definition is rather compact in that it omits any other aspects that a technology implicates. It lies in the argument's interest to expand on it so that "technology not only consists of the artefacts which are employed as tools, but also includes the sum total of the kinds of knowledge which make possible the invention, making and use of tools" (Gell, 1988). This definition does not only encompass the tool on its own but also addresses the knowledge that supports the tool. This is an interesting notion insofar that it introduces a socio-cultural element into an ostensibly technical domain. It becomes clear that the society from which a technology arises and the knowledge that surrounds it play an important role in the understanding and use of the technology. No technology exists on its own in a void but should ever be considered in a specific culturally defined light – albeit this belief is seldom addressed, as Bryan Pfaffenberger emphasises: "Technology, under the sway of Western culture, is seen as a disembodied entity, emptied of social relations, and composed almost entirely of tools and products" (Pfaffenberger, 1988). Hence, it is important for the framework of this argument that technology is instead understood as a vessel that is inhabited by a vast number of connotations, knowledge and social relations. It should be seen as "a system, not just of tools, but also of related social behaviours and techniques" (Pfaffenberger, 1988). Again, technologies should always be thought of as means originating from a societally influenced negotiation process attempting to tackle a given challenge.

Technologies and their development process are often seen as deterministic – technology as an ever-influencing powerful being – or "somnambulistic" – technology not meriting closer study; it suffices to "make it" or "use it" (Winner, 1986). Both views uphold the notion of the autonomy of technology relative to society while both of their roots can be found in the fetishisation of technology, a phenomenon that is commonly observed by anthropologists throughout human civilisation. For an object or a technique to appear as a fetish, it necessitates three attributes (although other sources state four), specifically "anthropomorphisation", "conflation of signifier and signified" (e.g., money, which only represents worth but is treated as if money itself possessed worth) and "an ambiguous relationship between control of object by people and of people by object" (Ellen, 1988). Karl Marx has applied and developed the concept of fetishism of commodities in a capitalistic system, while Pfaffenberger in turn sees fit to adapt Marx's argument to the technological domain, arguing that a technology is analogous to a commodity in that it is just as much part of the "mist-enveloped regions of the religious world. In that world the productions of the human brain appear as independent beings endowed with life, and entering into relation both with one another and the human race" (Marx, 1938). Technology's fetishised status as a mystified and inaccessible entity clouds the many influences that society exerts on it and vice versa. Thus, if technology should not be considered as deterministic nor somnambulistic, it becomes evident that choice and perspective significantly contribute to the development process of new technologies. If societal influences subsequently guide innovators in a given direction, it is within their power to question this specific direction and if deemed just, decide on a different one. David Noble explains that "[…] there is always a large measure of indeterminacy, of freedom, within it [the technological development]. Beyond the very real constraints of energy and matter exists a realm in which human thoughts and actions remain decisive. Therefore, technology does not necessitate. It merely consists of an evolving range of possibilities from which people choose" (Noble, 1984). As it stands, technology's "appearance […] of automaticity and necessity, though plausible and thus ideologically compelling, is false" (Noble, 1984). The element of choice from Noble's explanation should be considered essential to the overarching argument. Since it has been established that, firstly, there is no mystic facet in technologies that veils their functioning or their purpose and that, secondly, technologies, their logic and their functioning are not born out of pure necessity, choice appears to become a key influence.

The purpose of the preceding exposition is to showcase that the conception of a technology does not only result through an objectively determinable optimisation but also through numerous subjective stimuli, internalised societally defined thought patterns and indeed through choice. It should be briefly noted that even the purest aspiration to optimise is still a rather subjective desire as long as a human sets the boundaries of what optimisation implies, as the attribute of 'being optimal' is in itself part of an anthropomorphic view on the world (compare to the philosophical perspective in *2.4 The Anthropomorphisation of Failure*). In any case, what should be retained is that technology is an artefact comprised of uncountable influences and the distillate of choices made by its creators – a "hardened history, frozen fragments of human and social endeavour" (Noble, 1984).

Since the development of new technologies is thus based on societal influxes and individual choices, it should be investigated how some of these choices can have a negative effect on a technology's outcome, which, in the end, will give this paper the opportunity to scrutinise the current operating principle of GAs. As has been discussed, besides the number of actual limitations of a technology's physical framework that are stated by Noble, much of what constitutes a new technology is at its creator's discretion. At this stage, creators need to be aware of their personal biases, which might influence a number of decisions taken by the creator in a direction that might oppose its initial or actual objective. Biases originate through societal norms, personal experiences or various other influences and can harm the decision-making process in a an initially unperceived manner by arbitrarily neglecting scientific or functional evidence in favour of perceived truths or internalised customs. The scope of biases is probably best illustrated through the example of male bias in design and science, assessed by Caroline Criado Perez. In this case, the bias is highly developed in such a way that it has cultivated a male default. Perez demonstrates through numerous examples that "products marketed as gender-neutral that are in fact biased towards men are rife across the (male-dominated) tech industry" (Criado Perez, 2019). The resulting shortcomings resemble those of assistive fall detection apps, that have difficulties registering women falling as the algorithms are programmed to recognize falls through their accelerometer's data closer to the body's centre of mass while affected women most often store their smartphones in their purses (Criado Perez, 2019). Another example of male bias is the Virtual Reality's focus on the simulation of motion parallax instead of shape-from-shading to help the eye perceive depth.

Women depend primarily on shape-from-shading for depth perception, while men rely on motion parallax, which might be the reason behind the much higher number of women experiencing motion sickness in VR (Criado Perez, 2019). Other, more frightening examples stem from the medical domain, in which the effects of a given drug are insufficiently tested on women, ignoring the fact that possible side-effects in women often differ from those that male test subjects encounter during the trials (Criado Perez, 2019).

This brief foray into gender inequality in design and science intended to portray the obvious bias that is still a sadly peculiar and only slowly vanishing component of technological development, despite the fact that it affects half of the planet's population. The magnitude and vast propagation of this prevalent bias suggest the existence of countless more subtle biases that influence humans and algorithmic creators alike. One might think that machines would perform without these apparent shortcomings. However, as machines and algorithms are still conceptualised and trained by humans, their programmers' internalised biases are unconsciously passed down to their digital proxies and replicated by them (Shorter, 2020).

To summarise, it has been established that technologies are closely connected to society and that their mode of operation is seldom without alternative but has rather been defined by the technology's creator. Hence, it is entirely appropriate to question the choices made since they might be unintentionally interspersed with biases. In order to further the argument for controlled contaminations it should be asked which biases are exerted on GAs and whether the choices that were made during the conception of GAs could profit from more varied influxes.

The history of the GA's development clearly shows influences of the Darwinian evolutionary theory, a system that sought to understand how biological species were able to prevail through the course of time. The concept of survival of the fittest has at the very least become a common notion imprinted in the collective memory and has most certainly left an impression on the GA's selection method. Once more, it is essential to stress that, although Darwin's theory outlines the factual proceedings of evolution, the theory in itself – its connotations and the perspective from which it is written – withal bears the mark of human interpretation and is therefore qualified for detailed reassessments in relation to the process as such. That said, inspired by Darwin's findings, GAs work to eliminate a solution's traits that might be seen as unfavourable in respect to the algorithm's design goal. This modus operandi was arguably reinforced by the universal social bias that being wrong or mistaken is an undesirable occurrence. And yet, Kathryn Schulz considers the matter of being wrong to be more nuanced: "Paradoxically, we live in a culture that simultaneously despises error and insists that it is central to our lives" (Schulz, 2010). Although few people would claim they liked to commit mistakes, most would acknowledge that they learned from these mistakes. "Far from being a sign of intellectual inferiority, the capacity to err is crucial to human cognition. […] And far from being a mark of indifference or intolerance, wrongness is a vital part of how we learn and change" (Schulz, 2010). One might agree that GAs learn from their mistakes as well, in order to develop towards their optimum. But contrary to humans they forget past mistakes and discard the solutions that they deem flawed.

The anthropological perspective has shown that the functionality and the reception of technologies are highly dependent on the society which they originate from respectively are utilised in. The common bias that every kind of internal mistake is harmful to a technology's functioning has been exposed and can be regarded as disproven, at least in its most stringent and extensive form. "To create a new technology is to create not only a new artefact, but also a new world of social relations and myths in which definitions of what 'works' and is 'successful' are constructed by the same political relations the technology engenders. It could be objected, to be sure, that a technology either 'works' or it doesn't, but this objection obscures the mounting evidence that creating a 'successful' technology also requires creating and disseminating the very norms that define it as successful" (Pfaffenberger, 1988). Following Pfaffenberger's argument, the proposed controlled contaminations would hence not automatically render a GA's outcome defunct, but rather expand the definition of an optimal solution's boundaries and potentially alter the societal normativity of flawed design to a certain extent.

5.4 The preferences of the brain

The preceding chapters have treated the nature of erring and the differing opinions on what constitutes a functioning technology. Both perspectives offer a valuable insight into the repercussions of mistakes on human interaction and how society shaped the human perception of technologies and vice versa. Another field of interest to a GA's mode of operation touches on the domain of aesthetics. And while this subject has already been the matter of discussion in the previous philosophically influenced chapter, this section will cast light on neuroaesthetics, a field of study that explores the subconscious, biologically and psychologically motivated reasons for aesthetic judgement. Incorporating neuroaesthetics is justified by the close thematic relationship between an algorithm that emulates the evolution of species and the evolutionary development of the human brain's sensitivity towards beauty, especially since the latter's sensibility directly determines the output of the former. On the whole, neuroaesthetics deliver a scientifically verifiable understanding of a process whose workings are notoriously difficult to discern because of its predominant subjectivity on one side and the wide range of different aesthetics that exists – from landscapes and faces to music and modern art – on the other side. But looking at the brain like the electrochemical computer that it ultimately is, one is able to make a number of assumptions about the roots of aesthetic experience – some being more pragmatic, while others remain somewhat hazy. A short introduction into the field of neuroaesthetics serves as departure point that should provide a better understanding of the basic principles.

A substantial amount of neuroscientific research in relation to aesthetics focuses on vision, as it is the primary and most accessible sense capable of aesthetic experience (Starr, 2013). Neuroscientists divide vision into different subclasses that are being processed in separate parts of the brain. It is such that "form and colour are processed in one stream and tell us the 'what' of an object. Luminance, motion, and location are processed in the other and tell us the 'where' of an object" (Chatterjee, 2013). Without knowing it, artists have catered to these different sections of the brain with ever-changing art styles. They "are exploiting the characteristics of the parallel processing perceptual systems of the brain to create their works" (Zeki, 1998). Thus, art talks to specific regions of the brain. Most certainly, these brain regions were initially not adapted to the enjoyment of art. Instead, the visual apparatus has been trained through hundreds of thousands of years to respond to various stimuli that, when interpreted correctly, would give the beholder certain advantages in its fight for survival or when it was time to find a suitable mate for procreation. Classic art recreates those stimuli in a somewhat abstracted manner when it represents more or less realistic variations of the known world in the form of still lifes, landscape paintings, portraits etc. The underlying causative agent appears to be deeply etched into the human brain as different studies indicate. The 'savanna hypothesis' states that even people who have never visited a savanna prefer this kind of landscape when shown a set of different environments. This preference is especially prevalent in young children, but it is diluted the older the test subjects are in age, as this group starts to favour the surroundings, they have grown to know throughout their life (Balling & Falk, 1982). People have evidently learned to gain pleasure from the sight of landscapes that facilitates their survival. And while it may seem like this method follows an inevitably rigid pattern that is hard-coded into the human brain, unable to change, it has been established that aesthetic judgement is strongly influenced by the context in which the judgment is made as "we can change our emotional experience of objects by altering the context in which we consider them." (Chatterjee, 2013). Framing then becomes an important part to how humans classify objects; it "means that we are not slaves to our sensations. Our cognitive systems can reach down into our pleasure centres and rejigger our pleasure experiences" (Chatterjee, 2013). This process has in part been explained in the chapter about the anthropology of successful technologies. However, through the neuroscientist's perspective, unlike recognising society's influence, it is personal pleasure that is considered the root of the personalised aesthetic experience. To repeat: evolution was essential in defining at which sight pleasure should be felt, but the context in which something is seen is just as determinative as the inherent object.

At this stage, a first cautious relation between the human response to a GA's output and neuroaesthetics can be drawn. As has been established, when presented with an algorithm's solution, the human will assess its aesthetic value following various deep and unconscious processes, while at the same time taking into account the context in which this assessment occurs, namely at the end of a highly efficient optimisation process. This makes it entirely possible that the context overrules the primary aesthetic sensation (along the lines of: 'Such an optimised process must produce beautiful results.'). Going forth, one should bear this possibility in mind as a general reminder of the residual bias that invariably inhabits a GA's output.

With respect to the deeply embedded connotations of the natural environment Anjan Chatterjee argues that humans "are deeply integrated with the natural world" and that their "mind has been sculpted by nature and it is tightly coupled to the environment", only to pose the question of "what in the coupling of mind and world [it is that] gives us the experience of beauty" (Chatterjee, 2013). Since this question has already been answered to a certain degree with evolutionary reinforced pleasure, the issue becomes particularly relevant once modern abstract art is considered, since it does not necessarily evoke the lived environment. It stands to reason that the natural elements of landscapes abundant in nutrition or offspring-promising faces are seldom found in more abstract art. In many art forms, the lack of "primary reinforcers", whose "value is […] a priori predictable because these rewards have universal significance" (Starr, 2013) is a highly discussed topic throughout the neuroscientist field.

If not for lush environments and attractive faces, what then is considered beautiful by the human brain? Harry Mallgrave explains that "an abstract composition consisting of a few colours or forms may be processed only in the areas V3 or V4 [two areas of the brain that process different shapes], whereas a representational scene, as brain scans reveal, engages these areas as well as other parts of the cortex, no doubt soliciting memories or knowledge of previous experiences" (Mallgrave, 2009). Chatterjee comes to a similar conclusion and states that according to the "results of experimental neuroaesthetics studies, we find that there is no art module in the brain. […] The brain responds to art by using brain structures involved in perceiving everyday objects – structures that encode memories and meaning, and structures that respond to our enjoyment of food and sex." (Chatterjee, 2013). The perception of art as such has thus never been a primary skill of human perception. It has much more evolved through the combining of different brain regions that work together und produce the enjoyment of art. However, as shall be seen later, this does certainly not mean that today's understanding and dissection of art cannot be classified as an actual secondary capacity.

In the domain of architecture, with regard to Vitruvius' three virtues *firmitas*, *utilitas* and *venustas*, the question arises which factors play a role in the brain's enjoyment of architectural design. The neuroaesthetics of architecture is still only an emerging field due to the numerous stimuli and senses playing a part in the experience of an architectural space, making them difficult to assess. In contrast to the fine arts, where vision most often covers all the stimuli that are emitted by an œuvre, architecture exists in three dimensions, is smelled, felt, heard, and seen. For the longest time, the first two virtues of strength and utility have been scientifically quantifiable; only beauty has constantly been at the centre of controversy. In regard to architecture's current state of affairs, Chatterjee et al. maintain that the modernist philosophy "came to embody a new aesthetic ideal, reflecting a view of architectural beauty as nothing more than a byproduct of functionalist design", "while discarding long-observed aesthetic conventions like ornamentation and human scaling" (Chatterjee, Coburn, & Vartanian, 2017) , which is why they support a focused neuroaesthetic study of architecture. The researchers identify several key structures that are responsible for the experience of architecture: the sensory-motor systems, subdivided into vision, nonvisual experiences (e.g., olfactory, auditive and somatosensory) and motor responses, knowledge-meaning systems and emotion-valuation systems. Just like in art, vision is indeed the primary sense through which architecture is processed in various regions of the brain. During the appreciation of architecture, vision-related systems recognise various stimuli that are also seen in nature, like fractal geometry, symmetry (omnipresent in architecture due to the brain's efficient recognition thereof and "biomorphic features" with "nature-like visual qualities" which are seen "in the design of ornamentation, scaling, proportionality, and even structural support schemes" (Chatterjee, Coburn, & Vartanian, 2017) (Joye, 2007). Similar preferences also apply to art in general. The exact workings of these systems do not need to be explained in the context of this paper. It suffices to say that similar methods are applied by the brain when it engages with art or architecture, even though architecture requires additional capacities whose functioning should be explored in greater detail, for they might give valuable insights, potentially enabling improved designs. Furthermore, the existence of natural features in the (built) environment that are analysed by the brain on a deep level, suggests that there are indeed subtle elements in the architectural design process that are not necessarily consciously recognizable by the designer and possibly much less so by an algorithm that was programmed to optimise a design. While this realisation does not make a case for controlled contaminations in a GA, it hints at the possible shortcomings in the workings of a more standard algorithm.

Another noteworthy detail in the field of neuroaesthetics relevant to the functioning of GAs is the human response to average versus outlier features. Studies have shown that humans prefer faces with average features over faces that are more individual (Rubenstein, Kalanakis, & Langlois, 1999). This preference is outlined by Chatterjee through the greater genetic diversity that average faces imply, whereas a more diverse gene set correlates with a healthier organism. Chatterjee further cites the human brain's inclination toward prototypes as a reason for the preference of average faces, since the fast recognition of prototypes has made the human navigation of a complex world much easier, giving them an evolutionary advantage. Outlier features might hint at deformations or sickness, which is why humans preferably avoid them (Chatterjee, 2013). The population of a GA's optimisation process gravitates towards averageness, just as human faces do. As more and more flaws are eliminated, the last generations preceding the final output lose in diversity while increasing their fitness, in order to create a prototype that is best suited for the given design task. The human and the algorithmic proceedings are thus very similar. However, there is another element in human facial preferences that stands somewhat in conflict with the generally preferred averageness. The peak-shift principle denominates "an exaggerated response (the peak response) to exaggerated versions (the shifted version) of a stimulus that would evoke a normal response" (Chatterjee, 2013), meaning that certain features in faces, bodies or objects, but especially in sexually dimorphic features are displayed or formed in an exaggerated manner so as to cause an enhanced response on the observer's part. For instance, a very defined jawline is not average and yet it can enhance the attractiveness of its bearer. One can assume that this sort of outlier could very well be defined as a flaw by a GA. And yet, as has been established, if applied correctly, such a peak-shift outlier is able to elicit a positive reaction. A controlled contamination that simulates the peak-shift property could be a valuable addition to any design-oriented GA.

The final argument born of the insight into neuroscience is twofold and incorporates a songbird on one side and an architectural element on the other one. While expanding on the subject of the costly display (i.e., features of a species that, instead of holding an advantage, make the specimen's survival harder, correlating the specimen's overall fitness with the prominence of the feature), Anjan Chatterjee evokes an example that demonstrates the effects of "relaxation of selective pressures" (Chatterjee, 2013) (Snell-Rood, Van Dyken, Cruickshank, Wade, & Moczek, 2010). Bred and mated from the still existing wild munia with the intent to produce more vibrant feathers, the domesticated Bengalese finch has learned to sing a variety of sophisticated songs while being able to learn new songs based on the surrounding songs it hears. The ancestral munia still only knows its much simpler conventional song. Without the practical need for a cohesive song, "the natural drift and degradation of genes that program the stereotypic song could occur. The contaminated genes allow for neural configurations that produce songs that are less constrained and easily perturbed" (Chatterjee, 2013). This reconfiguration even affects the bird's brain structure so that when "genetic control over brain function got looser, instinctual constraints on the bird’s song got less specific" (Chatterjee, 2013). While some species' features are formed through high selective pressure as is the case with costly display and the modus operandi of GAs, a low selective pressure entails a different set of features or skills, like the finch's elaborate song or, as some researchers assume, the emergence of art in human societies. To summarize the principle of the two opposite evolutionary forces, Chatterjee explains that "typically, genetic mutations get weeded out. The evolutionary engine that does the weeding out drives behaviours toward uniformity, toward what evolutionary biologists call 'fixation'. When adaptive behaviours are relaxed from selective pressures, deviant behaviours do not need to be eliminated. The behaviour no longer matters for survival. It is free to 'mutate'. So, in the case of increased selective pressures, variety is culled; in the case of relaxed selection, variety blossoms" (Chatterjee, 2013). Hence, it is conceivable that a relaxation of the GA's optimisation process or the controlled introduction of the discussed contaminated genes into the population could be similarly advantageous in ways that need yet to be established.

The architectural element of the spandrel bears significance in evolutionary biology as it designates features that have developed because of internal necessities without being linked to an adaptation to given environmental influences, but in some cases have become functional at a later time (Gould & Lewontin, 1979). The link to architecture is drawn because of the spandrel's necessity as soon as a specific building form has been chosen, only to evolve into an installation of ornamentation and adornment. Steven Gould and Richard Lewontin name various kinds of seashells whose exoskeleton has formed patterns or grooves through the specific way that calcium is produced during growth. Uses for these patterns or grooves were secondary but became more pronounced once novel applications presented themselves. The proposed controlled contaminations could be established in a similar fashion. Although the actual need for such a deliberate flaw might not arise, the possibility might always occur eventually. Needless to say, this argument is not the strongest and possibly optimistic in its nature, but in the search for the GA's amended mode of functioning, every parallel to evolutionary biology should be considered, given the GA's functional proximity to its strongest inspirational source.

Since the aim of this chapter is to showcase the advantages of controlled contaminations in a process that is built on optimisation – a paradoxical endeavour – the step into the inner, deeper workings of the human brain and genetics seemed appropriate. After this foray, it becomes apparent that the manner in which the human brain processes its environment to compute its appreciation patterns is abundant with intricacies that hold a number of interesting lessons for how a modified GA could be designed.

5.5 The laboratory of purity

The notion of purity is a close companion to any optimisation process. In ridding a design of certain flaws that stand in the way of a predetermined outcome, the optimisation process creates a pure solution. While purity comes with a great variety of connotations it should be clarified that, in the context of this work, pure does not relate to any moral, ethical or religious beliefs. Purity designates much more the state of an optimised entity that, in relation to its intended purpose, is free from any flawing agent. Such is the basic principle of any given GA. The algorithm seeks to rake out as many components as possible that conflict with the given design goal. Its outcome is supposed to be as pure as possible with respect to said goal. However, purity and with it the very notion of an optimised solution it represents, is not a given, objective attribute, as has been stated previously in a similar manner. Purity's ambivalent disposition is best illustrated through the philosophy of chemistry's notion thereof.

As a scientific field, Chemistry has changed radically throughout the course of time. The comprehension of the chemist's work and the very essence of their studies has been subject to constant change and radical revolutions. From Empedocles' and Aristotle's ancient theory of the four basic elements of water, fire, earth and air with their respective states of dryness and humidity, hotness and coldness, which they believed were the ingredients of all matter, to the entrance into modern science and the "analytic era" (Bachelard, 1972) that has been greatly influenced by Antoine Lavoisier, chemistry has undergone many transformations in the way it approaches its object of investigation as well as how its sister sciences perceive chemistry. These changes are accompanied by the quest to rise from the condescending denotation of an impure science to the ranks of a precise one that is capable of postulating "the first principles for the science [of chemistry] or elaborating general laws" since chemists had merely been able "to prove tentative, restricted generalizations derived from an experimental process of trial and error" (Bensaude-Vincent & Simon, 2012). Many of these past transformations have been caused by scientific discoveries that led to a redefinition of what chemistry was actually investigating.

The pursuit for the homogenisation of all matter was arguably one of the most important steps toward the status of a recognised science. Successfully breaking down the complicated mixtures into their simple components promised to finally enable the analytical observation of matter's disposition. This was on no account a trivial enterprise, with Pierre-Joseph Macquer stating that "the simpler any substances are, the more sensitive and considerable are their affinities; whence it follows that the less the bodies are composed, the more difficult it becomes to analyse them, specifically to separate from one another their composing principles"[1] (Macquer, 1756). It is this practice of uncovering pure elements in combination with the development of intricate tools capable of determining these elements in order to "guarantee the quality and purity of a variety of products" that would help chemistry "in rising to the social status that they enjoyed in the second half of the eighteenth century" (Bensaude-Vincent & Simon, 2012). Through countless trials and experiments – practical ones, as chemistry has always been an applied science, expanding its knowledge by means of erring; much unlike its historically more widely respected sister science, physics, which has profited from a distinctly theoretical branch – chemists had worked out ways to discern the different components of a given substance in order to produce molecularly pure materials. This work made chemists coveted and granted them respect since the beginning of the industrial age when the demand for such purified materials grew rapidly.

One should consider examining the notion of purity in this regard. As Bachelard notes, "one cannot avoid recognising that the *purity of substances* belongs to the *human realm*. It does not belong to the *natural realm*. In fact, the human is the

[1] Transl. "Plus les substances sont simples, plus leurs affinités sont sensibles et considérables ; d'où il suit que moins les corps sont composés, plus il est difficile d'en faire l'analyse, c'est-à-dire de séparer l'un de l'autre les principes qui les composent." p.22

purifying factor"[2] (Bachelard, 1972). While some native elements can be found in nature, it is in their designation of being pure, that Bachelard sees the human factor. The production of a new, pure element seemed unachievable at the beginning of the 18th century as Bachelard cites Johann Friedrich Henckel, who, at the time, declared that one would never achieve to decompose a *mixture* without creating a new one; the attempt to look at a substance's principles separately and in their simplest state would always be in vain (Bachelard, 1973). Despite that, technological advancements made this ostensibly impossible task a reality. It is vital stress that a pure substance is not a s*imple* substance: "The simple is not a given but the result of a secure homogenisation technique. […] Purity, in the realm of matter, is never a mere given as much as it can never simply be detected. At least, it cannot be detected without using a method of experience, which, in modern science, summarises an antecedent chemical rationalism. […] An *impure* body is in short "tested" by bodies that are postulated as pure. There lies an effective dialectic which, through the course of the scientific ages, demonstrates the progressive determination of purity, without purity ever being able to establish itself as a certain given or an absolute"[3] (Bachelard, 1972). Since the notion of pure substances can only be found in the human realm and since the classification process is guaranteed by substances that are deemed pure themselves, it is evident that the entire notion of purity is a self-sustaining concept and may only ever be seen as an indicator of the present knowledge. Bensaude-Vincent and Simon say that "the concept of purity in this context is both provisional and conventional, as it is the chemist's tests and criteria that establish the yardstick of purity, and nothing, in principle, could prevent them from coming under attack" (Bensaude-Vincent &

[2] Transl. "on ne peut manquer de reconnaître que la *pureté des substances* appartient au *règne humain*. Elle n'appartient pas au *règne naturel*. C'est l'homme qui est en fait le facteur purificateur." p.89

[3] Transl. "Le simple n'est pas une *donnée* mais le *résultat* d'une technique de sûre homogénéisation. […] la pureté, dans le règne de la matière, n'est jamais proprement une *donnée*, qu'elle ne peut être même simplement *constatée*. […] En tout cas, elle ne peut être constatée sans mettre en action une méthode d'expérience, méthode qui, dans la science moderne, résume tout un rationalisme chimique antécédent. […] Un corps *impur* est en somme « essayé » par des corps qu'on postule *purs*. Il y a là une dialectique agissante qui marque, au long des âges scientifiques, la progressive détermination de la pureté, sans que jamais cette pureté puisse se signaler comme une donnée sûre, comme un absolu."

Simon, 2012). This leaves purity as a concept in a highly subjective and temporally obscure domain.

The chemist's laboratory is the situated space of any purification process, "a place deliberately isolated from the rest of the world, and so has little in common with it. Yet, it is a place intended to generate truths about this same natural world" (Bensaude-Vincent & Simon, 2012). The same can arguably be said about GAs, as they build their designs in a simulated space that stands in place for the natural world in which the designs are intended to fulfil their task. The algorithm's purification process is a different one since it does not entail a simplicity of characteristics but a congruous complexity of concurrent requirements. And yet there is a remarkable parallel in their respective approaches beyond the basic principle of an optimisation-related purification. Similar to technology in its sociocultural dimension, Bruno Latour emphasises the fetishisation of the purified elements. The fetishes that result from a purifying transformation of natural elements become artefacts, supporting "the robust certitude that permits the movement from practice to action without ever believing in the difference between construction, collection, immanence or transcendence" (Latour, 1996). The purification and in consequence the fetishisation of natural elements is a necessary step in the scientific process as it guarantees an objective and undimmed analysis of the elements at hand but comes at the cost of leaving something behind – in this case the elements' history of coming to be, one that is unthinkable without the reliance on social conventions (cf. Bachelard: "the social aspect of the materialist inquiry"[4], "nothing proves the imminent social character of contemporary science better than purification techniques. In fact, the purification processes can only establish themselves through the utilisation of an entire aggregate of reagents whose purity has acquired a social guarantee"[5] (Bachelard, 1972)). While there can be no scientific progress without the required objective perspective, the origins of the purified object should still be of interest, even more so in more creative fields. If it is assumed

[4] Transl. "l'aspect social de l'enquête matérialiste"
[5] Transl. "rien ne peut mieux prouver le caractère éminemment social de la science contemporaine que les techniques de purification. En effet, les processus de purification ne peuvent se développer que par l'utilisa- tion de tout un ensemble de réactifs dont la pureté a reçu une sorte de garantie sociale"

that a purification process converts their solution into a fetish, it is evident that the thusly formed fetish loses the knowledge that constituted the solution prior to its purification and that "the resulting laboratory substance is completely different from the original, inevitably idiosyncratic natural substance. Stripped of its natural history, the circumstances of its production and the techniques that intervened in its extraction, the 'raw material' has been transformed into a chemical species; a pure material abstraction" (Bensaude-Vincent & Simon, 2012). The abstraction of a given substance is not to be considered a simple "aestheticization of the experiment that turns the laboratory into a closed chamber in which entirely autonomous things happen"[6] as Hans-Jörg Rheinberger and Michael Hagner clarify in their work about experimental systems; the process shows much more "what produces and reproduces itself, stabilises and becomes unstable, deforms and reforms itself"[7] (Rheinberger & Hagner, 1993). This confirms that the laboratory is a space in which scientific processes are paramount and with that it should be stressed that the removal of a substance's "natural history" is indispensable in the realm of chemistry and therefore not subject to criticism. However, it is a circumstance that should always be taken into consideration, especially in the context of GAs. Remnants of its creation process are arguably an important component of any creative design, be it in the form of a first pencil sketch underneath a classic oil-painted portrait or the obvious formal influences of an iconic armchair; it is certainly not intuitive to strip a creative design of its history. GAs (or their programmers) should be aware of the entailments of purification processes.

Now that the utility of chemical purification has been illustrated, the subsequent re-complexification should not be left unexplored. While the purified elements allow to draw conclusions about their characteristics, it is mostly in combination with other elements that they are able to fulfil a practical purpose. Oftentimes, the coalescence of a primary homogenous element with a minor amount of another element will drastically alter its characteristics and qualities. Gaston

[6] Transl. "Ästhetisierung des Experiments zu lesen, die das Labor zu einer verschlossenen Kammer macht, in der gänzlich autonome Dinge geschehen"
[7] Transl. "sich produziert und reproduziert, stabilisiert und instabil wird, sich deformiert und reformiert"

Bachelard illustrates this facet with an example of the purity of a hydrocarbon liquid that "possesses an electrical conductibility which varies from 10^{-19} mho/cm in an extremely purified sample to 10^{-13} mho/cm in a commercial sample, meaning a variation of 1 to a million", with the help of which "we can see the enormous repercussions of the most minor impurity"[8] (Bachelard, 1972). Although Bachelard refers to a substance that varies in its degree of purity because of purification processes of varying rigour, the same principle applies to deliberate blends, for instance semiconductors. Semiconductors are constituted of an element whose electrical conductibility increases or decreases in relation to the material's temperature. Used as a gate on a MOSFET transistor, the semiconductor's conductibility is controlled via an applied voltage, whereby it will act as a switch between a source and a drain. For this purpose, the semiconducting element (most often Silicon) needs to be built in layers of n-type and p-type semiconductors. For n-type semiconductors, "this is achieved by doping, i.e., contaminating, the silicon crystal with a pentavalent element such as phosphorus during production [...]. This means that some silicon atoms are replaced by phosphorus atoms in the crystal lattice, whereby one bond of each of the pentavalent phosphorus atoms in the crystal lattice remains unsaturated"[9] (Göbel, 2019). P-type semiconductors are doped with a trivalent element, like boron. "The boron atom acts as a so-called acceptor in the crystal lattice, i.e., it very easily accepts a fourth electron in the crystal lattice"[10] (Göbel, 2019). These different atomic builds allow the passing of electrons from the source to the drain through the gate, enabling the MOSFET transistor to be used as a logic gate in most modern electronics. In the context of this paper, the focus should be placed on the aspect of contamination. Only through contaminating a pure silicon crystal with an exact atomic dose of another

[8] Transl. "un hydrocarbone liquide a une conductibilité électrique qui varie de 10^{-19} mho/cm pour un échantillon extrêmement purifié à 10^{-13} pour un échantillon commercial, soit une variation de 1 à un million. On voit l'énorme action de la moindre impureté."
[9] Transl. "Dies lässt sich dadurch erreichen, dass bei der Herstellung der Siliziumkristall mit einem fünfwertigen Element wie z. B. Phosphor dotiert, d. h. verunreinigt wird [...]. Dies bedeutet, dass in dem Kristallgitter einige Siliziumatome durch Phosphoratome ersetzt werden, wobei jeweils eine Bindung der fünfwertigen Phosphoratome in dem Kristallgitter ungesättigt bleibt."
[10] Transl. "Das Boratom wirkt dabei im Kristallgitter als so genannter Akzeptor, d. h. es nimmt im Kristallgitter sehr leicht ein viertes Elektron auf."

element can modern transistors be built on a microscopic scale and deliver the performance that is needed for efficient computing. The argument of this paper is that the same type of controlled contamination might have benefits for design-oriented GAs: a minimal and precise intervention that confers new capacities to the aggregate, even though these capacities may not be immediately ostensibly noticeable.

If a GA's purification process is a distillation of a design's purest features, one should bear in mind that "water used as a solvent in the chemistry laboratory should not taste like Malvern water, Mediterranean water or water from Evian. It should have no taste at all, it should be “pure” water – generally regarded as unfit for human consumption" (Bensaude-Vincent & Simon, 2012).

5.6 A different aesthetic aspiration

A final foray into the extensive world of aesthetics leads to the Japanese aesthetic ideology of wabi sabi. Over the centuries, its two constituting words have continually increased in nuance. Etymologically, wabi stands in connection with "languish, […] loneliness, forlornness, and wretchedness" (Juniper, 2003), which, different from what one might expect, has no pejorative connotation, but means to evoke a simple, antimaterialistic life, reminiscent of Zen Buddhism's teachings, from which wabi sabi draws its main inspirations. Sabi searches "to convey a sense of desolation, […] of utter loneliness and finality […] and it went hand in hand with the Buddhist view on the existential transience of life" (Juniper, 2003). Combined, wabi sabi is evocative of an aesthetic that embraces impermanence, flaws and "the imperfections found as all things, in a constant state of flux, evolve from nothing and devolve back to nothing" (Juniper, 2003). With these characteristics, wabi sabi is in stark contrast with some western aesthetics that often rely on symmetry, persistence and purity.

The principles of wabi sabi have been applied to flower arranging, traditional gardening, tea ceremonies etc., but it is in the art of ceramics where they are most intricately illustrated. Through ash glazing in sophisticated furnaces, Japanese craftsmen are, to a certain degree, stripped of control when it comes to the determination of the ceramic product's definite appearance. The colourful, textured appearance of the resulting artifact is a powerful witness of the flux of the hot gases and the ashes originating from the large amount of softwood burnt in the furnace. This process ultimately leaves no room for any optimisation on the part of the craftsman. If the ceramic's aesthetic "is supposed to be perfect in its form and glaze, then, apart from the inevitable flaws that it will have, there will be less to hold the attention" and in creating "pieces that may appear physically imperfect, the artist is offering an opportunity to get involved in the piece and to help complete the picture, or to even reflect on the seemingly imperfect nature of life itself" (Juniper, 2003). The tenet of demanded introspective is just as true for traditional Japanese gardening. Open spaces, which, through a western perspective, could be characterised as voids, and natural rocks placed in seemingly random but in truth carefully considered locations and groups engage with their beholder by challenging to seek greater truths in the garden's composition.

The inherent quality of the aesthetic seems obvious and offers a tantalizing alternative to the prevalent approaches of the west that have followed similar patterns since Greek antiquity. Wabi sabi's principles are apparently in diametrical opposition to the optimisation process of a GA – it appears counterintuitive to charge an optimisation algorithm with generating flawed solutions only to evoke the transience of all things. And yet, it is despite a similar counterintuitive facet that Jean-Francois Maheux sought to unify mathematics with wabi sabi. In his paper *Wabi-Sabi Mathematics*, Maheux argues that even a field as concerned with "qualities such as inevitability and non-triviality" in which "most of those who reflect on the aesthetics of mathematics focus on elements such as symmetry and simplicity" (Maheux, 2016), may find new stimuli through the ideals of the Japanese aesthetic. Maheux states several examples that are meant to illustrate the wabi sabi nature of mathematics: the "ambiguity of mathematical ideas", mathematics' imperfect disposition "in the sense that not every mathematical statement can be proved to be true or false", the "beauty in the observations that straightforward or precise answers cannot be reached", as well as "the fact that some problems can be solved only through 'rough' calculations that often remain practical approximations" (Maheux, 2016). An advantage of seeing mathematics through a wabi sabi perspective might be found in the manner in which mathematics are taught to students: to rethink mathematical studies as a perfectly imperfect enterprise in order to alter the conception of mathematics' immutable objectivity (which seems legitimate if one considers that there is indeed still unclarity about what a number actually is) (Maheux, 2016). Maheux rightly asks why one should celebrate one's flaws only to conclude that their acceptance is vital to reconcile one's inevitable erring with mathematics' "often non-functional, contradictory, uncontrollability, organic face" (Maheux, 2016).

As such traits are far from being exclusive to mathematics, Hans-Jörg Rheinberger and Michael Hagner argue that "following the tracks of the sciences' articulation, it becomes apparent that epistemological activity cannot be reconstructed adequately unless the unwanted, the unknown and the blurry are also left that space to which, according to classical conception, science as a rational and predictive enterprise appeared to be appointed"[11] (Rheinberger & Hagner, 1993). The same is arguably true for design-oriented GAs, as perfect optimisation is unachievable in the light of the often contradictory, uncontrollable, and organic nature of its solutions.

In a more practical approach to wabi sabi, Francesca Ostuzzi et al. have researched imperfections in industrial mass-production to conclude that "the research has seen the value of imperfection as an economic opportunity for the industry; the strategy proposed here is to exploit the flaw as a generator of unique products and inspiration of the mass customization for users" (Ostuzzi, Salvia, Rognoli, & Levi, 2011). The study demonstrates that a reconsideration in terms of an appreciation of the flawed might even have economic benefits (even though this paper would not argue for the wilful production of generic flaws with the aim of increasing profits by conveying a false sense of uniqueness).

[11] Transl. "Den Fährten der Artikulation von Wissenschaftsdingen folgend, zeigt sich, daß Erkenntnistätigkeit nicht angemessen rekonstruiert werden kann, wenn nicht auch dem Ungewollten, dem Ungewußten und dem Unscharfen jener Raum belassen wird, aus dem herauszuführen Wissenschaft als rationales und prädiktives Unternehmen nach klassischer Vorstellung gerade berufen erschien."

While wabi sabi does not endorse any principle similar to controlled contaminations, the acceptance and valorisation of the inherently flawed disposition of human creativity should be an inspiration to the updated GAs this paper is arguing for.

6 Controlled Contaminations

The previous chapter has portrayed the various fields that might stand in as role models for the integration of controlled contaminations into GAs. The ways in which these fields employ, are subjected to or speak for controlled contaminations varies greatly, which should not come as a surprise given their diversity. It is evident that most of these arguments are not directly linkable to generative algorithms. Despite that, they demonstrate the significance that the flawed has throughout all these disciplines in relation to human creation and culture. This, along with the previous critique of prevalent aesthetic judgement, should suffice to justify the experimental step of flawing an optimisation in order to create a stronger connection between algorithmically generated designs and humans by whom they are beheld or employed. This concise section will introduce the proposed contaminating function.

6.1 A cautious delineation

Controlled contaminations appear in all sorts of forms, depending on the observed field: sometimes in the form of ashes, adorning a ceramic bowl with a distinct texture, other times as chemical elements, granting electronic components distinct properties, or even as intentionally mutated genes that alter an organism's genetic blueprint. The repercussions of the contaminating interventions vary in form and magnitude. Universally defining controlled contaminations is challenging due to their multifaceted applications and requirements. It is the hope of the author that the reader has gained a precise enough insight into the nature of contaminations through the examples that have been provided in the previous chapters. This work will briefly attempt to define how such a contaminating principle can be established in the programming of GAs, where they are supposed to act as a simple, yet elegant function that intervenes during the optimisation process. Because of the overwhelming diversity of design tasks, this work is unable to define a universally applicable function that can be utilised in any algorithm, as a sort of plugin. It should be explained that this is not a potential shortcoming, but much more a principle that is grounded in the necessity for universal customisability. A principle that is just as true for the definition of the already established fitness criterium, a task that requires a revaluation for the computation of every new optimisation problem.

Nevertheless, it is entirely possible to outline a few best-practice criteria that are informed by the findings of the previous chapters and that should help to program algorithmic controlled contaminations in a manner that ensures an effective application.

- Depth of the contamination's influence:
  As mentioned in *5.1 Mistakes, explained*, contaminations should never hinder a design's primary functionality, while surface algorithmic flaws are deemed desirable. For instance, a contamination shall not affect a building's statics by deforming supporting columns. Instead, it could offset a window in an otherwise symmetric window distribution.
- History:
  A contamination should consist of an element that has been discarded due to the algorithm's optimisation process. This principle is evocative of the abstract significance, which can be attributed to a chemical substance's provenance, as explained in *5.5 The laboratory of purity*. Choosing discarded elements from the design's past distinguishes a contamination from the process of mutation, which can be characterised as a blind altering of the design's features that seeks to produce new influx.
- Randomness:
  The decision about which elements among the design's flawed features are chosen should be a random process. Similar to the creation of wabi sabi ceramics (described in *5.6 A different aesthetic aspiration*), the elements of uncontrollability and coincidence should be central to the concept of contaminations.

- Appropriately timed application:

  Although not rooted in the preceding findings, it is however vital to define when contaminations may take place during the algorithm's execution. Considering that contaminations will likely lower a phenotype's fitness, it is expectable that the algorithm discards the contaminated phenotype during the next selection process. To avoid the detection of the contamination, it should only be applied after the fitness calculation and before the selection process. This way, the phenotype is assessed in its original state through the fitness calculation and is later passed into the crossover function in a contaminated condition so that it can pass on its contaminated genes.
- Storage:

  Even though the contaminated phenotype is enabled to produce offspring after having avoided the fitness calculation, its offspring is likely to be discarded in the following generation. It is therefore advised to store the offspring in a separate, contaminated population for later retrieval and usage. A second separate optimisation run with the members of the contaminated population should increase their fitness while keeping elements of the contaminations in the final output.

The criteria defined above are not to be understood as necessary features of a contaminating function, but rather as guidelines that have proven beneficial during tests with the proposed function in the context of the experiment in the following chapter.

7 Empirical study of a contaminated GA

With the aim of showcasing controlled contaminations, a GA has been programmed that follows the postulated principles. Not thought of as an undisputable proof of the superiority of its solution, the algorithm was conceived as an intricate exploration of applied controlled contaminations.

Written with this purpose in mind, the program explores the possibilities of generatively shaped urban plans. Instead of creating detailed, infrastructurally sound maps, it draws inspiration from the concept of field conditions to produce more artful, yet functionally informed arrangements. Before the experimental algorithms and the achieved results can be discussed, a brief introduction into field conditions, as well as the algorithm's application of the theory are necessary to gain a better understanding of the chosen framework. It should be stressed that the chosen subject of generatively created urban plans is neither essential, nor directly linked to the aim of this paper. It serves as a mere illustration of the proposed principle of controlled contaminations and was chosen on the basis of criteria that render the demonstration feasible in the scope of this paper.

7.1 Field conditions

Proposed by Stan Allen, the method of field conditions was conceived to offer a different approach to the analysis and understanding of architecture and urban fabrics.

> "Field conditions moves from the one toward the many, from individuals to collectives, from objects to fields. In its most complete manifestation, the concept of field conditions refers to mathematical field theory, to nonlinear dynamics, and to computer simulations of evolutionary change. […] Field configurations are loosely bound aggregates characterized by porosity and local interconnectivity. Overall shape and extent are highly fluid and less important than the internal relationships of parts, which determine the behavior of the

field. Field conditions are bottom-up phenomena, defined not by overarching geometrical schemas but by intricate local connections. Interval, repetition, and seriality are key concepts. Form matters, but not so much the forms of things as the forms between things. Field conditions cannot claim to produce a systematic theory of architectural form or composition. The theoretical model proposed here anticipates irrelevance when faced with the realities of practice. These are working concepts derived from experimentation in contact with the real" (Allen, 1997).

Allen postulates that the relationship between entities – e.g., buildings in a city, rooms in a building, or columns in a room – and the space that these relationships employ, bear the central importance as to arrange things accordingly. The relationships are not informed by hierarchical geometries, but by the "effect emerging from the field itself" (Allen, 1997), which is to say that relationships and geometries are formed by their constituents and not vice versa. Allen draws inspiration from the artist Barry Le Va's work, whose terminology of "distributions" is borrowed by Allen as he evokes Le Va's denominations for his respective work in his explanation of what a field is characterised by: "Whether 'random' or 'orderly' a 'distribution' is defined as 'relationships of points and configurations to each other' or concomitantly, 'sequences of events' […][, so that every design should] 'transcend its first appearance of disorder to another level of order…. When chance methodology is used extensively enough it does not necessarily produce a disorderly or accidental-appearing distribution'" (Livingston, 1968).

The local connections that are examined by the field conditions approach are convenient to the GA's iterative mode of production. The entities that contribute to the field's appearance or use are a bottom-up phenomenon, which allows the algorithm to produce local mistakes i.e., local formations that could be described as functionally redundant (e.g., two fire stations on the same block). However, in doing so, they do not disrupt the sort of overarching rationale that would be found in classic urban planning, because there is no sort of an organised "masterplan".

As Allen himself admits, field conditions lose their relevance once they are applied to practical city planning. Even so, they enable the artful study of cities through a different lens by highlighting intricate relations. While these insights do not inform an immediate application, they could give planners "new definitions of 'parts,' and alternative ways of conceiving the question of relationships among those parts" so that "a space is left for the tactical improvisations of future users. A 'loose fit' is proposed between activity and enclosing envelope" (Allen, 1997). It is the inherent vagueness paired with the functional, albeit at times turbulent composition of the urban fabric that make field conditions a fitting framework for a GA that, on the one hand, strives to optimise and on the other hand explores the effects of deliberate mistakes through controlled contaminations.

7.1.1 Field conditions in the context of the proposed algorithm

The field is understood as an unbound accumulation of locally connected entities. The form of those entities is insignificant, while the relationship they share is the primary building block of a field's quality. The distribution within the urban playing field is an additional essential property. The exact nature or the qualifications of these connections or the prerequisites for an embedding into a distribution are not precisely defined by Allen. Their affiliation may however bear resemblance to the qualities of a flock as it is described by Allen: "The flock is clearly a field phenomenon, defined by precise and simple local conditions, and relatively indifferent to overall form and extent" (Allen, 1997). Taking these key attributes into account, the algorithm analyses data of existing cities to subsequently rebuild the city in various configurations while respecting a certain set of features that are observable in the actual model. To achieve this, arrays of urban infrastructure nodes are extracted from the city map to form specific fields. A field's expanse and influence grow in parallel to the connections that are established by the entities it internalises. Whether or not two nodes form a connection depends on the distance that separates them.

For a better understanding of the proposed principle, one can picture a lone university building situated in a given urban context. The building is likely to have a certain influence on the surrounding infrastructure, as the students' daily needs can be a catalyst for the emergence of an adapted infrastructure in the vicinity of the building. Public transport, restaurants and other economies will adjust to the local requirements. The presence of several university buildings in the immediate and intermediate vicinity, entailing an increased number of students, will expand the influence the buildings exert on their neighbourhood. A different illustration of this principle are the possible effects of police stations on their surrounding area.

By building and visualising these fields of influence and their formative nodes, it is possible to paint an alternative map of the city that, although omitting a large amount of information, finds value in portraying the distribution of the different infrastructures of the layered fields. The deconstruction of a given city into the parts that spring from a field conditions analysis and its subsequent reconfiguration by a GA can be understood as an exploration of the city's inherent relationships detached from historic and geographic stipulations. Aspiring to optimise the reconfiguration in the image of the existing city which is specified as a relative optimum in the way that it is a functioning urban disposition – albeit never optimal in its own right – in a first step, the algorithm will proceed to test out the controlled contaminations. Re-injecting the discarded mistakes it made during the optimisation process into an otherwise optimal solution will diminish the solution's fitness in relation to the optimum. At the same time however, the procedure is likely to create a number of novel local relations. A visual comparison of the different results will make it possible to judge the effectiveness of controlled contaminations in the context of field conditions in the exploration of the urban fabric.

7.2 The algorithm's approach

With the purpose of creating alternative city maps based on the field conditions analysis of an existing city, the experiment's setup consists of three separate programs: the data mining algorithm that draws the real-world geolocation of a city's existing infrastructure from a mapping service, the central genetic algorithm that creates and optimises different maps based on the mined data, and lastly the visualiser program, that generates intricate abstract maps.

7.2.1 <u>The visualiser program:</u>

Although the visualisation program is the chronologically last component of the experimental setup, it seems advisable to explain its functionality first in order to provide a better understanding of what the remaining two programs are working towards. Assessing real-world data from the data mining tool or outputs created by the GA, the visualisation program draws two-dimensional field conditions inspired maps. Below is an example of a generated visualisation.

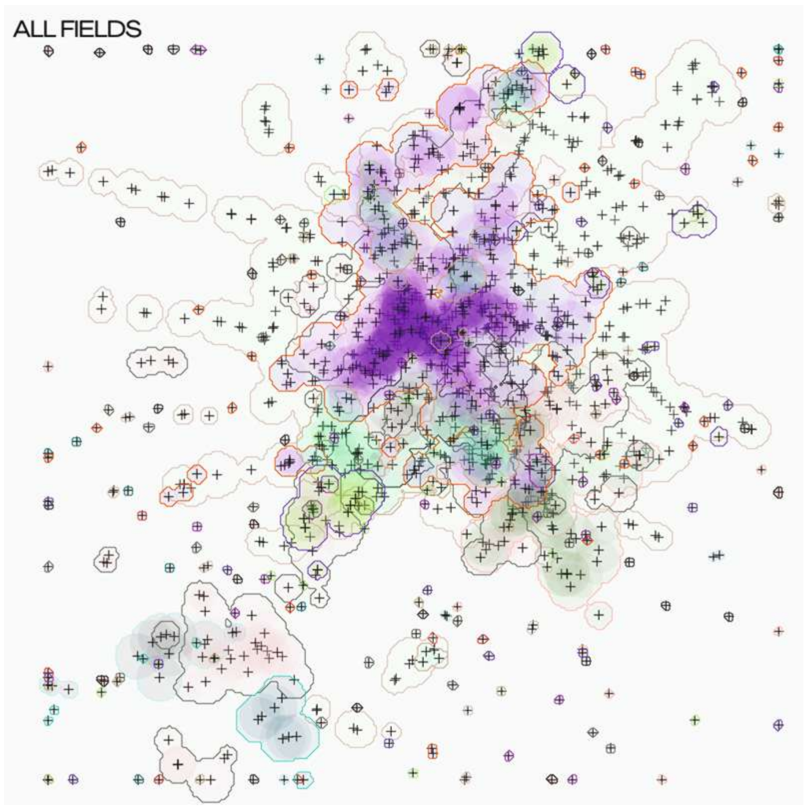


Figure 4: Fieldmap view of all the fields combined

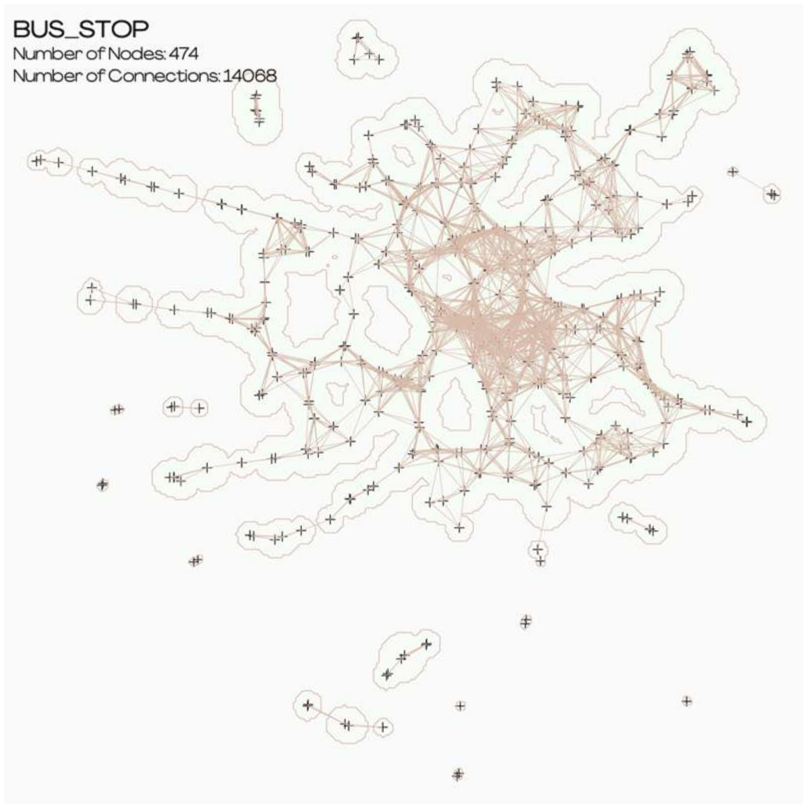


Figure 5: Fieldmap view of a single field with visible connections

The two images shown depict the two views of the visualised map: the first draws every generated field on top of each other, while the second view only displays one specific field. The name and a numerical context are given in the top of each window. Every map consists of the following elements:

- *Nodes*: each node, in the form of a black cross, represents a member of a field, i.e., one infrastructural element.
- *Connections*: nodes that lie within a given distance from another node form a connection. Connections are depicted as coloured lines between nodes.
- *Circles of influence*: The coloured shapes surroundings each node are their circles of influence. These shapes grow in expanse relative to the node's number of connections. This principle has been explained in *7.2. Field conditions in the context of the proposed algorithm*. It is the visual representation of a field's extension.

The emphasis of the visualisation lies on the infrastructure's underlying distribution, its concentrations, and its nexuses. Omitting most of the information that constitute a classic map, the fieldmap gives the beholder a different perspective on a city's relationships.

It should be noted that there is no interaction between the different fields. Although this could be considered a useful feature, it was discarded for the time being, due to two non-negligible difficulties: first, the additional computational power required for cross-analysing different fields and second, the extensive theoretical work of defining which fields might interact with each other during the preliminary stages - an extensive task due to the large number of available categories and resulting combinations and the user's choice of which ones they choose or omit in their analysis.

7.2.2 The data mining tool

A compact Python application makes use of OpenStreetMap's Overpass API to gain the necessary information for the generation of urban maps. The inputs necessary for a search query consist of a city's name and corresponding country, as well as the API-specific designations of the infrastructures that are deemed interesting for the field conditions analysis. Drawing the requested data in the form of the infrastructure's lateral and longitudinal location from OpenStreetMap's database, the program exports CSV-files with every entry's name and coordinates that can then be processed by the genetic algorithm. The tool is designed in a manner that enables the user to easily customise it, depending on their specific requirements when analysing different cities, as the properties of different urban plans vary greatly, e.g., some cities maintain public ferries, while others employ cable cars in their public transportation system. The data mining tool is a separate application and functions independently from the GA.

7.2.3 The genetic algorithm

The central constituent of the experimental setup is the genetic algorithm. The Processing Development Environment was chosen as the designated programming tool, due to the writer's experience with PDE, as well as its ability to efficiently visualise the processed data. The superior efficiency of a Python-based GA was taken into account during the choice of the programming environment but was ultimately dismissed in favour of the inherent streamlining advantages of the GA and the visualiser program being written in the same language.

In a first step, the algorithm analyses the input data it receives from the data mining tool. The data obtained thusly is organised into the following data structure: every infrastructural element informs a *node*, which is subsequently grouped into its corresponding node array, a *field*, which, in turn, is embedded into a *fieldmap* containing every created field in an array of fields, one for each infrastructural category. A city's fieldmap is the field conditions-inspired representation of a city's urban framework.

The imported real-world dataset and a myriad of computed variables based on the dataset are stored by the GA as the optimum – a functioning city that stands as a model to the algorithm's generated solutions.

In a second step, the GA initializes a population of fieldmaps, by generating a random number of nodes for each field and placing them randomly inside the available space. The GA proceeds by repeatedly assessing the fieldmaps' fitness, combining their chromosomes relative to their respective fitness scores, and mutating the offspring until a certain fitness is reached. A high fitness score signifies that the generated map fulfils the required criteria to qualify as a valid alternative to the model city in the context of a field conditions analysis. For a deeper understanding of the process, the three recurring steps of fitness calculation, crossover and mutation shall be explained in greater detail.

The fitness score calculation of each fieldmap is based on the key feature of every field's point cloud ratio, which is calculated separately for every field contained in the fieldmap: the point cloud ratio denotes the ratio between the areal occupancy of the nodes' circle of influence of points, that is growing with each established connection (as described in 7.2.), to the total number of occupiable points in a superposed imaginary, regular point grid.

The average of every field's ratio is returned as the fitness score for each fieldmap. Relative to their scores, each member of the population is added n-times to the mating pool, i.e., a higher fitness score renders the random crossover selection between two high-fitness mating pool members more as they are overrepresented in the mating pool compared to lower fitness members.

The crossover phase combines the data of two parent fieldmaps into a child fieldmap, through a single point crossover during which the first half of every field's nodes of the first parent is passed on to the child, only to be completed by the second half of every field's nodes of the second parent.

For the mutation of the child fieldmap the method of an Adaptive Genetic Algorithm (AGA) inspired by Zhu et al. (Zhu, Lee, & Wang, 2021) was chosen. The AGA implements an adaptive mutation rate which is determined by an equation that factors in the child's separately calculated fitness score, the population's average fitness and the error between the average fitness and the optimal fitness. An adaptive mutation rate is helpful in avoiding premature convergence onto local optima and maintaining a high diversity throughout the population. In general, lower fitness scores entail higher mutation rates that insert different influxes into the population while higher fitness scores profit from lower mutation rates that, in general, seldom distort nearly optimal solutions. In mutating the child, the AGA randomly chooses from a range of possible measures while iterating through every node of every field:

- delete the node
- change the node's position to a most likely different random position
- add an additional random node to the node array

These three steps ensure the ongoing diversification of the population's chromosomes by altering the fieldmap's building blocks on the lowest possible scale – the number and location of their constituents. After mutating the child, it is added to the new generation of the population. The preceding generation is deleted to free up computational memory.

The procedure is repeated until a user-defined number of generations has been completed. Once the set target has been reached, the best performing fieldmap's data is exported in the form of CSV-files which can then be imported to the visualiser program for analysis and for comparison to the map's real-world model.

7.2.4 The contamination

The setup that was explained to this point describes the functioning of a standard GA (with a few characteristics of an AGA) and the proposed contaminant function has thus far been left out. The purpose of the standard algorithm without the contamination is that of a control group during the experimental setup. Its results are used in a comparative manner when assessing the value of the contamination's effects. The contamination's implementation is direct and relatively simple in its approach and extent. After having assessed the population's fitness during the fitness calculation stage, the algorithm chooses two members with the highest or the lowest fittest score respectively. These members' genotypes are subsequently passed into a contaminant crossover function which, in contrast to the standard crossover function, does not split the partners' genetic material in half to create their offspring, giving both partners an equal chance of passing on their genes. The contaminant crossover function favours the fitter member's genotype by passing it on in its entirety. In a second step, through the application of the algorithm's mutation rate probability, the function substitutes some of the newly created offspring's genes for genes that stem from the second partner with the lowest fitness score. The contaminated child is then stored in a contaminated population which exists in parallel to the standard population. The algorithm's final solution is then picked from the pool of contaminated solutions by assessing the population's fittest member. The uncontaminated phenotype is exported simultaneously from the standard population, so that both versions can be compared.

The purpose of implementing the contamination after the fitness calculation and storing the contaminated offspring in a separate population is to allow them to circumvent the following generation's fitness calculation's discarding function, since the contaminated solutions will have a lower fitness score than their non-contaminated originals. By doing so, the contaminated members are kept safe from a probable elimination and can be accessed at the end of the program, when their fieldmap's data is exported in the form of CSV-files, similar to the standard GA's procedure.

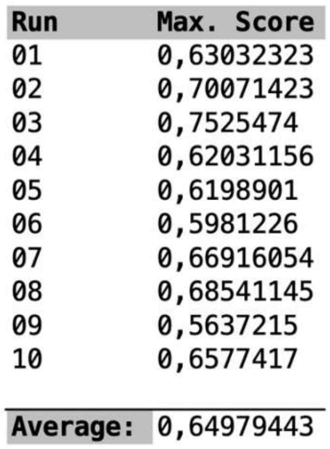
This paper's initial intent to single out and at a later stage reintroduce specifically marked mistakes, i.e., certain phenotypes' features that are determined to be flawed, has proven to be difficult to implement in the chosen setting of field conditions, as the underlying interwoven data structures are too complex and their features too inexpressive to be individually analysed and potentially categorised mistakes.

7.3 Experimental setup

During the empirical study, the results of the standard GA trained on generating fieldmaps based on the analysis of an existing city are compared to the city's original fieldmap layout and the solutions produced by the same GA with the addition of the proposed contaminating function. Ten runs of each algorithm, standard and contaminated, are performed and end after 20 generations. The achievement of a particular fitness score was considered as the algorithm's stopping criterion but was discarded over time, as test runs were showing that the algorithm's solutions tended to vary in quality without attaining the initially desired fitness scores. The population size is set to 1000 phenotypes and thereby strikes a compromise between a high enough diversity upon initialisation and computational efficiency, as populations become more computationally intensive as they increase in size. The described setup in combination with the representation of seven infrastructural categories resulted in a computation time of approximately 1,5 hours per run. The selected seven categories to be represented in the generated output, exported through OpenStreetMaps from the city of Salzburg, Austria, are the following: café, church, commercial, hotel, police, school, theatre. Salzburg was chosen among several cities as it was considered to be of fitting size, so as not to overcrowd the generated map exhaust the computational capacities; evidently, any city of comparable size could have been chosen just as well. The choice of categories is based on their diverse nature as well as the number of nodes each category contains. Some interesting and informative categories (e.g., bus stops) were pondered to be included, but had to be dismissed due to the large number of nodes they contain (there are 472 bus stops), which tended to result in suboptimal fitness scores during the test runs. The final maps – whose nodes' coordinates are exported to CSV-Files and subsequently imported into the visualiser program – are saved as image files for analysis.

7.4 Results

After the completion of the Field conditions GA's ten population's optimisation runs, the following fitness scores have been obtained. Each score belongs to the fittest uncontaminated fieldmap of the 20th generation. Its contaminated counterpart exhibits the same score as the fitness calculation takes place before the contamination.

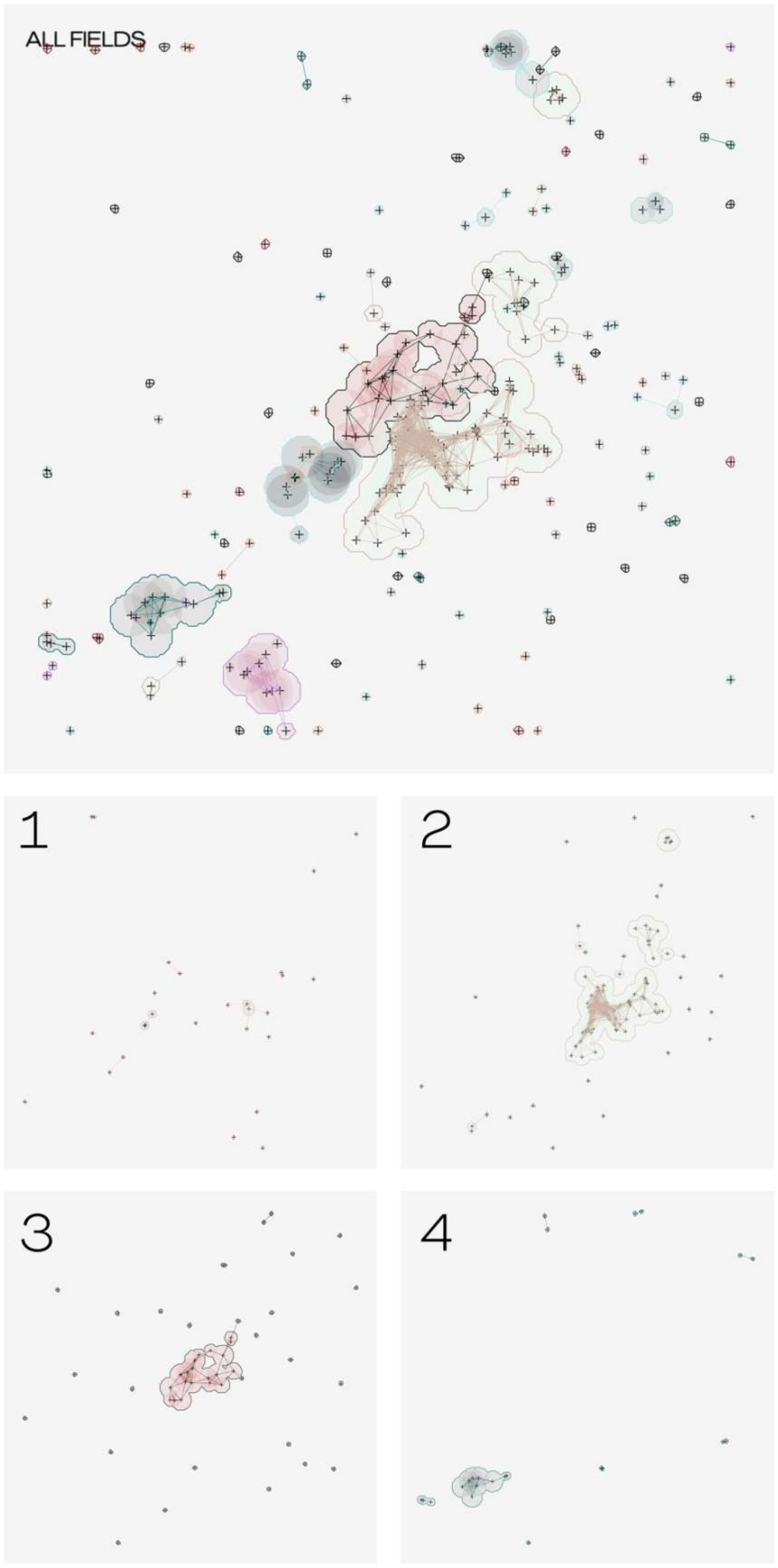


Run	Max. Score
01	0,63032323
02	0,70071423
03	0,7525474
04	0,62031156
05	0,6198901
06	0,5981226
07	0,66916054
08	0,68541145
09	0,5637215
10	0,6577417
Average:	0,64979443

Table 1: Fitness scores of the ten experimental runs

Judging the quality of the generated output based on its fitness averaging on ~0,65, it can be discerned that the present program is not yet fully capable of producing fieldmaps that represent an existing city's features in a satisfactory manner, as values above 0,85 had been set as a desirable target. Still, it is entirely possible to analyse the algorithm's solutions – especially the differences between the contaminated and the uncontaminated phenotypes. Run 3 with a fitness score of 0,75 will be observed in a more detailed manner, while an additional map can be found in Appendix B.

The first map shown shows the data as it was extracted from OpenStreetMaps and serves as a reference for the generated maps.


ALL FIELDS
1
2
3
4

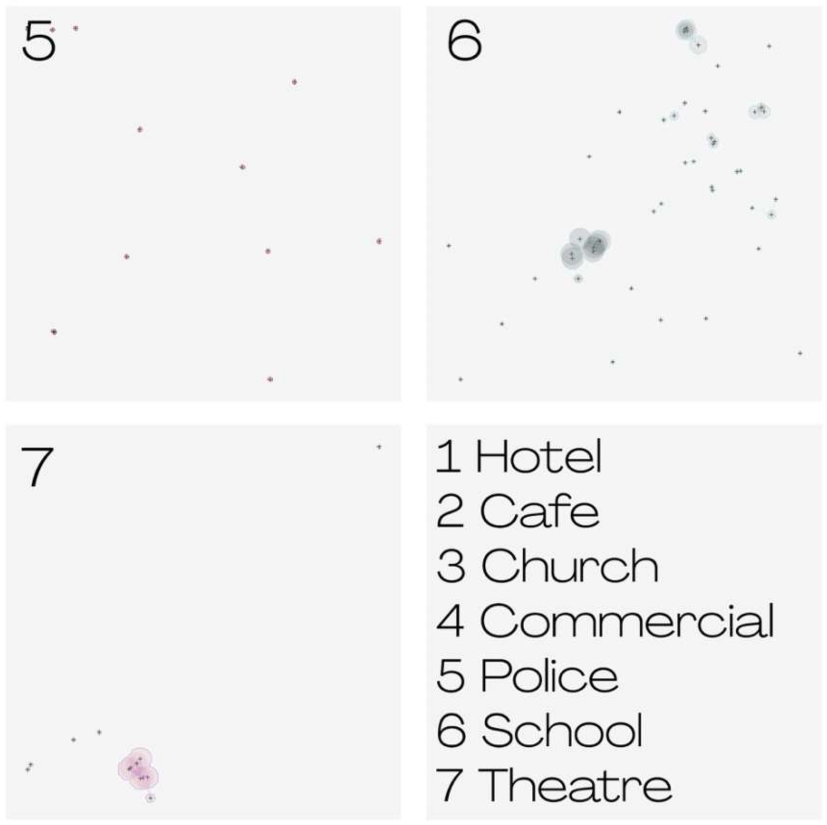


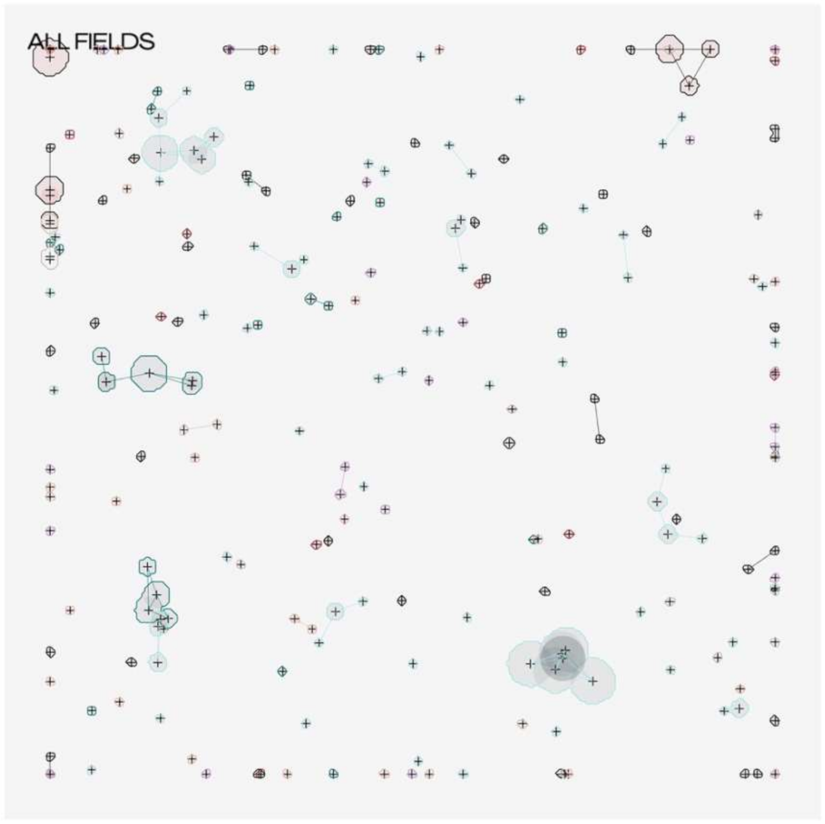
Figure 6: Optimum fieldmap

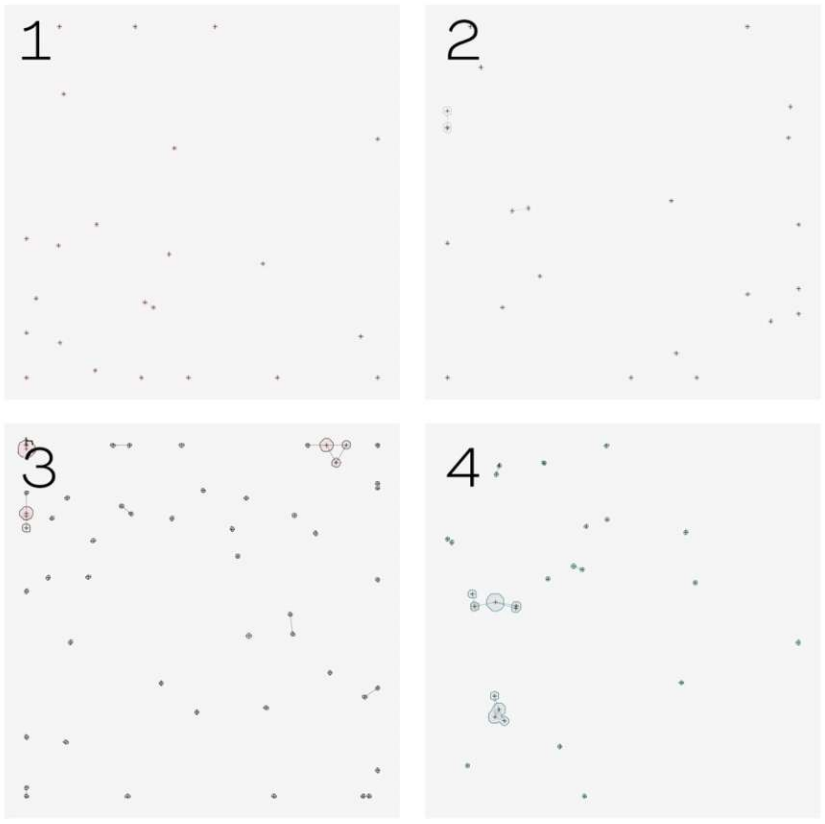
Field Name	Nr. of Nodes	Nr. of Connections
Hotel	27	11
Cafe	117	851
Church	49	48
Commercial	23	29
Police	12	1
School	58	91
Theatre	13	24

Table 2: Numerical data of the optimum fieldmap

The city's original fieldmap displays several distinctive field clusters, especially noteworthy in the domains of cafés and churches, which seems plausible considering the use and history of the infrastructure. Other fields, like those representing police stations and hotels, are more thinly spread over the map. In the case of police stations, this appears reasonable as this sort of infrastructure is built in a way that is supposed to provide an evenly organised coverage of its service. While the obvious infrastructural centre lies in the middle of the map, some smaller centres appear on the map's periphery – which is most notably for the commercial field, which might hint at the existence of bigger shopping malls.

Having provided the reference map of the existing city, the uncontaminated fieldmap of run 3 shall be analysed next. It should be mentioned that this analysis leaves room for interpretation in relation to significance of the generated features. When examining a fieldmap, one's mind should always try to imagine a sprawling, living city.


ALL FIELDS


1
2
3
4

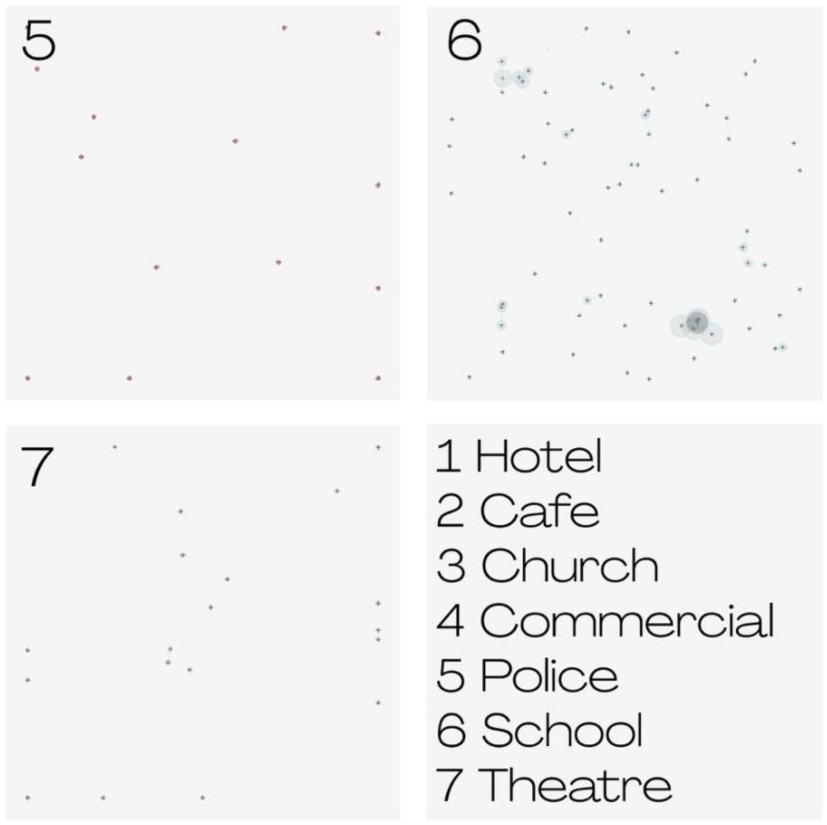


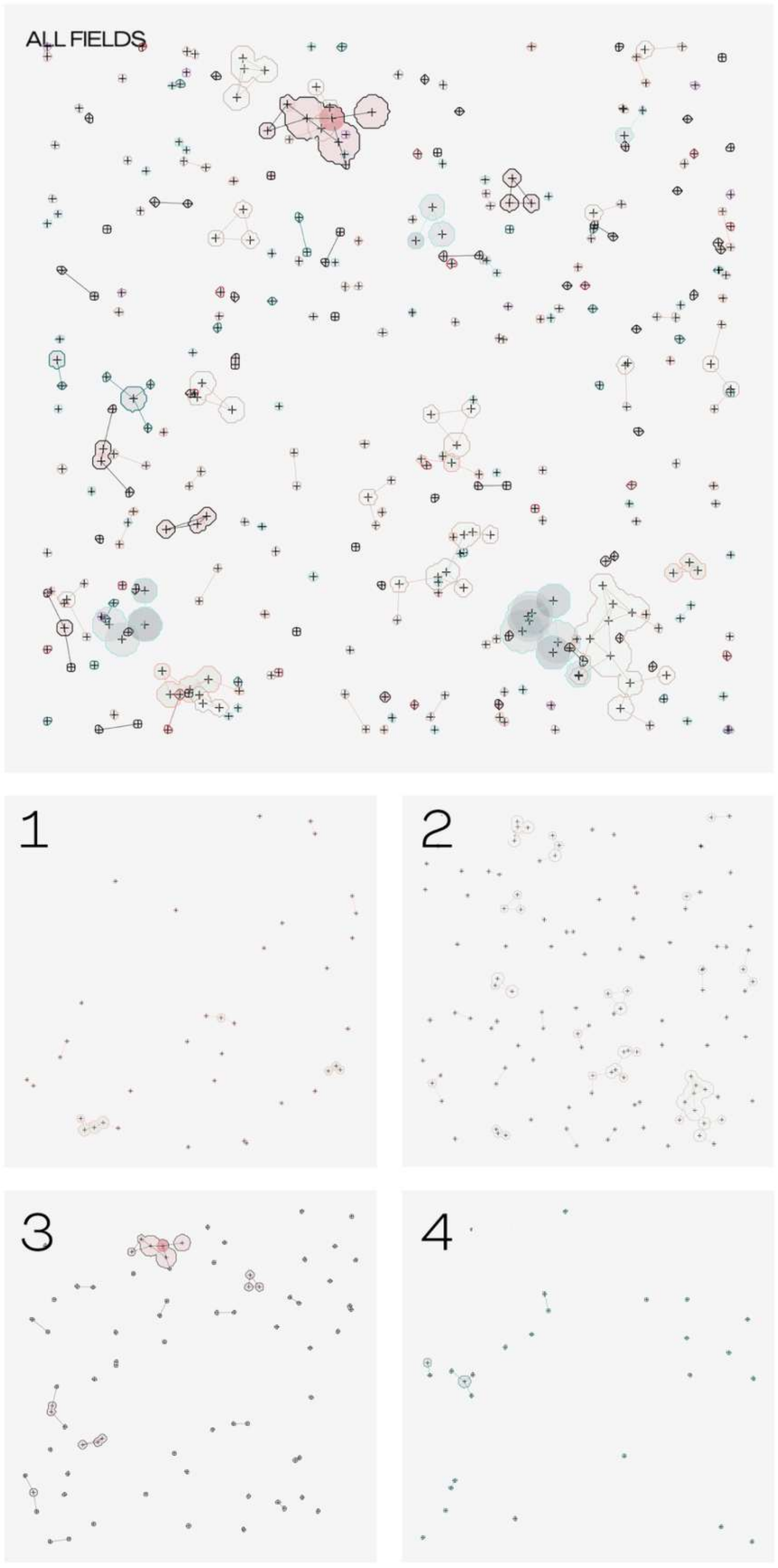
Figure 7: Run 3 uncontaminated fieldmap

Field Name	Nr. of Nodes	Nr. of Connections
Hotel	24	1
Cafe	24	4
Church	55	19
Commercial	28	15
Police	15	0
School	75	52
Theatre	21	2

Table 3: Numerical data of run 3 uncontaminated fieldmap

Run 3 has the highest fitness score of the experimental setup and yet, it appears to be organised in an entirely different manner than the fieldmap rooted in reality. As stated in the experiment's framework, the algorithm's fitness calculation checks the occupation of points in the map's underlying regular point grid by the nodes' fields of influence. The generated map might have found it beneficial to spread out its infrastructure more evenly throughout the map. In comparison, this solution leads to a comparatively low number of nodes and connections, which is especially true for the field of cafes. The only notable clusters are those of the churches, commercial buildings, and schools. This development shifts the city's infrastructural spike in leisurely activity to a much more even distribution, as most of the various categories' numbers have not changed significantly. Altogether, this solution does not appear favourable although it features the highest fitness score.

Following this relatively unsatisfactory fieldmap is its contaminated counterpart.


ALL FIELDS
1
2
3
4

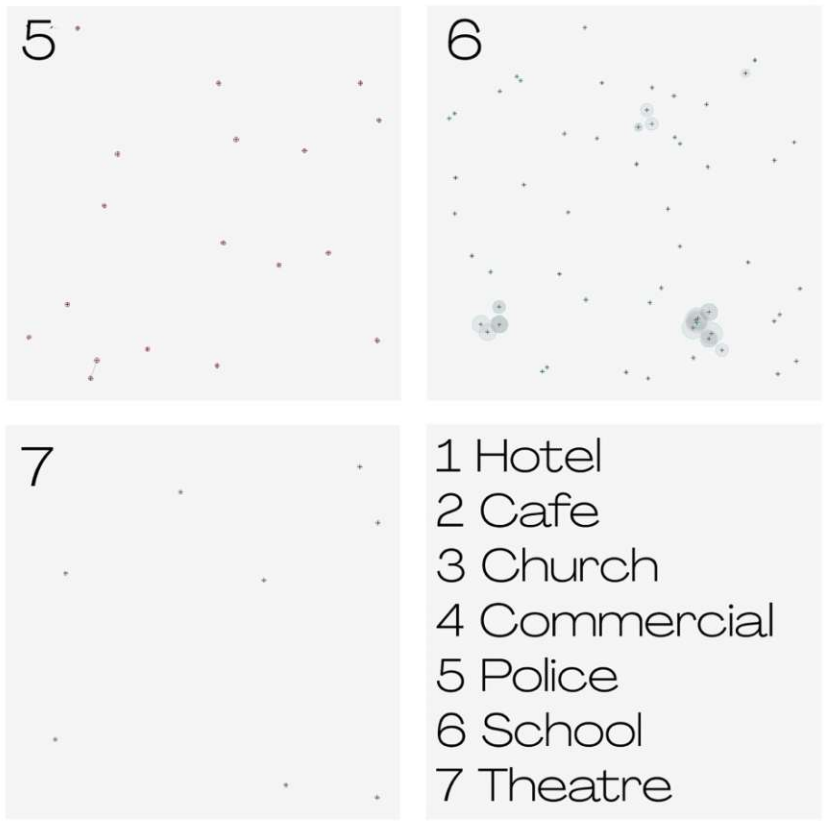


Figure 8: Run 3 contaminated fieldmap

Field Name	Nr. of Nodes	Nr. of Connections
Hotel	38	16
Cafe	125	81
Church	83	39
Commercial	28	7
Police	20	1
School	74	53
Theatre	11	0

Table 4: Numerical data of run 3 contaminated fieldmap

While the contaminated fieldmap's numerical data changes only slightly in relation to its uncontaminated self (with the exception of the café field), the overall visual impression is differing considerably from the standard GA's solution. The evenly scattered nodes have in part been clustered together, which enabled the emergence of more distinct centres. Hotels are still scattered but seem to form small groups around some points of interest. The higher number of cafes creates several compact fields, which can be seen an improvement over the standard GA's output. These smaller centres offer an intriguing alternative possibility in relation to the actual city's central concentration. The clustering of the churches appears to have improved as well, as some bigger centres might hint at the existence of a larger religious congregation. Commercial buildings remain relatively dispersed and form only smaller fields, which is also true for the generated police stations. Schools are also scattered throughout the map, but the existence of proper campuses is noteworthy. There are less theatres present in the contaminated fieldmap than in its uncontaminated and real counterparts, which leads to a noticeable and unfortunate lack of clustering. The superposed fields create the image of a ring-like city with de-centralised clusters of different activities that appears to be pleasing on a subjective level – more so than the product of the standard GA.

It can be said that the contaminated phenotype's fieldmap has seemingly led to an improvement of the map's quality. An additional, visualised fieldmap, whose fitness score of ~0,598 lies below average, can be found in Appendix B, so as not to further disturb the reading flow.

Overall, it becomes apparent that the chosen experimental setup of a fieldmap generation is not entirely suitable for the analysis of the proposed controlled contaminations' effects on optimisation processes and that no further meaningful information can be gathered from the fieldmaps in their current state of development.

8 Conclusion

Originating from the most inconspicuous fragment of a famous work by Alan Turing, sprung the idea of a particular addition to the discourse of genetic algorithms, whose concentration on uncompromising optimisation, was called into question. The more anthropomorphous aspect of mistakes in the form of subtle incisions into an algorithm's optimisation process is an unusual proposition in the sense that it explicitly calls for a less optimised optimisation. These incisions were proposed to be executed in the form of controlled contaminations.

In a first instance, Giorgio Agamben's seminal work, *L'uomo senza contenuto*, was consulted to provide the defining groundwork in the argument for controlled contaminations, using it as a medium to question and recast common conceptions in art and art criticism with a focus on algorithmically generated art, which invokes novel complexities such as the discussed unification of artist and art critic. It became clear that the common concepts of art vs. non-art and failure, as well as the relationship between a piece of art and its history could be redefined, which broadened the discussion around mistakes in the context of optimisation processes.

Subsequently, different, scientific disciplines, as well as an aesthetics ideology, were consulted to further define controlled contaminations and support their implementation. Turing himself has indirectly delivered the argument for the proposed kind of concept and the members of the domain of anthropology have argued in favour of embracing errors committed in the strive for innovation. Neuroaesthetics plead for the subtleties of aesthetic pleasure that might escape a classically culling algorithm, for the importance of the peak-shift-principle, which renders standout features more attractive than average features, and, ultimately, for the unforeknown advantages of relaxed selective pressure. Chemistry teaches of the impediments that have come along with the historical homogenisation of substances and the strive for purity, as well as the inherent benefits of contaminated materials and their subsequent increase in properties. A final argument is made by the Japanese aesthetics of wabi sabi that embraces the unforeseen imperfections of human-made designs as an organic part of their manifestation.

Bearing all these arguments in mind, it can be concluded that mistakes, errors, imperfections etc., appear to be a natural and sometimes even vital part of human creational endeavour. Optimisation processes, such as the ones employed by GAs might consequently necessitate comparable flawed elements. The introduction of the proposed controlled contaminations is hence a first step towards a more humane optimisation process.

At this point, it should be mentioned that the process of intentionally flawing a design with the aim of enhancing it in the perception of its user can be perceived as an optimisation in itself. It is an issue that does not immediately affect this paper's position and yet, the matter necessitates further attention. Should mistakes indeed enhance the quality of algorithmically generated solutions as is postulated, it would position contaminations in the ranks of an optimisation. On one hand, this circumstance should render their introduction far less controversial but on the other hand it puts into question the self-concept of the flawed, seeing that a modification can hardly be flawed if it ultimately optimises an outcome. It is an issue that cannot be solved in the scope of this paper but is nevertheless worth raising.

The empirical study, which marks the first implementation of the newly defined controlled contaminations and having been conceived to highlight the benefits thereof, was not entirely successful in its effort. While a functioning GA has been programmed which delivers intriguing results, these results do not promote the impact of controlled contaminations in a satisfactory manner. The generated fieldmaps offer fascinating insights into an unconventional kind of urban analysis and are the foundation for future projects that involve the algorithmic generation of similarly divergent city plans.

Future work might involve a more precise definition and mode of implementation for the postulated contaminations as they have only been outlined in vague and explorative manner. The presented fieldmap generator might as well be developed in a way that produces more potent solutions. Its integration into CAD programs or game engines seems advisable as this would enable it to produce three-dimensional and organically growing cities.

It is the belief of the writer that, based on the aforementioned arguments, controlled contaminations are an important evolution in the field of genetic algorithms and whose study should therefore be continued. This work is supposed to act as a starting point and the proposed controlled contaminations as a tool for designers and artists that wish to drive the field of algorithmic design forward.

9 Bibliography

Darwin, C. R. (1859). *On the Origin of Species.* London: John Murray.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* Cambridge: MIT Press.

Makki, M., Showkatbakhsh, M., Tabony, A., & Weinstock, M. (2019). Evolutionary Algorithms for Generating Urban Morphology: Variations and Multiple Objectives. *International Journal of Architectural Computing, 17*(1), 5-35.

Frankish, K., & Ramsey, W. M. (2014). *The Cambridge Handbook of Artificial Intelligenc.* Cambridge: Cambridge University Press.

Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind, New Series, 59*(236), 433-460.

Nisztuk, M., & Myszkowski, P. B. (2019). Hybrid Evolutionary Algorithm applied to Automated Floor Plan Generation. *International Journal of Architectural Computing, 17*(3), 260-283.

Manurung, R., Ritchie, G., & Thompson, H. (2012). Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence, 24*(1), 43-64.

Harding, J., & Brandt-Olsen, C. (2018). Biomorpher: Interactive evolution for parametric design. *International Journal of Architectural Computing, 16*(2), 144-163.

Kramer, O. (2017). *Genetic Algorithm Essentials.* Cham: Springer.

Mitchell, M. (1999). *An Introduction to Genetic Algorithms.* Cambridge: A Bradford Book The MIT Press.

Agamben, G. (1995). *Homo Sacer : Sovereign Power and Bare Life.* Stanford: Stanford University Press.

Agamben, G. (1999). *The Man Without Content.* Stanford: Stanford University Press.

Perlis, D., Purang, K., & Andersen, C. (1998). Conversational Adequacy: Mistakes are the Essence. *International Journal of Human-Computer Studies, 48*(5), 553-575.

Luger, G. F., & Chakrabarti, C. (2017). From Alan Turing to modern AI: practical solutions and an implicit epistemic stance. *AI & Society, 32*(3), 321-338.

Dennett, D. C. (1991). *Consciousness Explained.* New York: Little, Brown & Co.

Geiger, B. (2000). Discipline and classroom management in K thru 8th grades. *Education, 121*(2), 383-393.

Schön, D. A. (1967). *Technology and Change.* New York: Delacorte Press.

Gell, A. (1988). Technology and Magic. *Anthropology Today, 4*(2), 6-9.

Pfaffenberger, B. (1988). Fetishised Objects and Humanised Nature: Towards an Anthropology of Technology. *Man, New Series, 23*(2), 236-252.

Noble, D. F. (1984). *Forces of Production: A Social History of Industrial Automation.* New York: Knopf.

Ellen, R. (1988). Fetishism. *Man, New Series, 23*(2), 213-235.

Winner, L. (1986). *The Whale and the Reactor: A Search for limits in an Age of High Technology.* Chicago: The University of Chicago Press.

Criado Perez, C. (2019). *Invisible Women.* New York: Abrams Press.

Shorter, A. (2020, 4 7). *Bias in Machine Learning.* (Microsoft) Retrieved 2 19, 2021, from https://devblogs.microsoft.com/premier-developer/bias-in-machine-learning/

Neilson, J. (2012). Can Moral Flaws Count as Aesthetic Virtues? *The Journal of Value Inquiry, 46*(1), 65-81.

Chatterjee, A. (2013). *The Aesthetic Brain: How We Evolved to Desire Beauty and Enjoy Art.* New York: Oxford University Press.

Starr, G. G. (2013). *Feeling Beauty: The Neuroscience of Aesthetic Experience.* Cambridge: The MIT Press.

Zeki, S. (1998). Art and the Brain. *Daedalus, 127*(2), 71-103.

Balling, J. D., & Falk, J. H. (1982). Development of visual preference for natural environments. *Environment and Behavior, 14*(1), 5-28.

Mallgrave, H. F. (2009). *The Architect's Brain: Neuroscience, Creativity, and Architecture.* Chichester: Wiley Blackwell.

Joye, Y. (2007). Architectural Lessons From Environmental Psychology: The Case of Biophilic Architecture. *Review of General Psychology, 11*(4), 305-328.

Rubenstein, A. J., Kalanakis, L., & Langlois, J. H. (1999). Infant Preferences for Attractive Faces: A Cognitive Explanation. *Developmental Psychology, 35*(3), 848-855.

Snell-Rood, E. C., Van Dyken, J. D., Cruickshank, T., Wade, M. J., & Moczek, A. P. (2010). Toward a population genetic framework of developmental evolution: the costs, limits, and consequences of phenotypic plasticity. *BioEssays, 32*(1), 71-81.

Gould, S. J., & Lewontin, R. (1979). The spandrels of San Marco and the Panglossian paradigm: A critique of the adaptationist programme. *Proceedings of the Royal Society of London. Series Biological Sciences, 205*(1161), 581-598.

Bensaude-Vincent, B., & Simon, J. (2012). *Chemistry: The Impure Science.* London: Imperial College Press.

Macquer, P.-J. (1756). *Élémens de chymie théorique. Nouvelle Édition.* Paris: Jean-Thomas Hérissant.

Bachelard, G. (1973). *Le Pluralisme Cohérent de la Chimie Moderne.* Paris: Librairie philosophique J. Vrin.

Latour, B. (1996). *Petite Réflexion sur le Culte Moderne des Dieux Faitiches.* Paris: Synthélabo.

Juniper, A. (2003). *Wabi sabi: the Japanese art of impermanence.* North Clarendon: Tuttle Publishing.

Maheux, J.-F. (2016). Wabi-Sabi Mathematics. *Journal of Humanistic Mathematics, 6*(1), 174-195.

Ostuzzi, F., Salvia, G., Rognoli, V., & Levi, M. (2011). The Value of Imperfection in Industrial Product. *Conference on Designing Pleasurable Products and Interfaces.* Milano.

Rheinberger, H.-J., & Hagner, M. (1993). *Die Experimentalisierung des Lebens: Experimentalsysteme in den biologischen Wissenschaften 1850/1950.* Berlin: Akademie Verlag.

Marx, K. (1938). *Capital* (Vol. I). London: Allen & Unwin.

Bachelard, G. (1972). *Le Matérialisme Rationnel.* Paris: Les Presses universitaires de France.

Nietzsche, F. W. (2008). *The Gay Science.* Cambridge: The Cambridge University Press.

Benjamin, W. (1996). *Walter Benjamin: Selected Writings, 4: 1938–1940.* Harvard: Harvard University Press.

Schulz, K. (2010). *Being Wrong.* New York: Ecco Press.

Chatterjee, A., Coburn, A., & Vartanian, O. (2017). Buildings, Beauty, and the Brain: A Neuroscience of Architectural Experience. *Journal of Cognitive Neuroscience, 29*(9), 1521–1531.

Göbel, H. (2019). *Einführung in die Halbleiter-Schaltungstechnik.* Berlin: Springer Vieweg.

Allen, S. (1997). Field Conditions. *Architectural Design*, 24-31.

Livingston, J. (1968). Barry Le Va: Distributional Sculpture. *Artforum*, 50-54.

Zhu, Y., Lee, K. Y., & Wang, Y. (2021). Adaptive Elitist Genetic Algorithm With Improved Neighbor Routing Initialization for Electric Vehicle Routing Problems. *IEEE Access*, 16661-16671.

10 Appendix A: Code of the employed Programs

10.1 OpenStreetMaps_Exporter1_0 (Python)

```
from OSMPythonTools.nominatim import Nominatim
from OSMPythonTools.overpass import overpassQueryBuilder, Overpass
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
import pandas as pd

nominatim = Nominatim()
overpass = Overpass()
min_max_scaler = preprocessing.MinMaxScaler()

areaId = nominatim.query('Salzburg, Austria').areaId()
print(areaId)
# use 'place' and 'city' to get center coordinates of city
searchSelector1 = "railway"
searchSelector2 = "station"
#######################################################
# Use this method to get amenities
#query = overpassQueryBuilder(area=areaId, elementType='node',
selector='"amenity" = "place_of_worship"', out='center')

# Use this method to get building types
# query = overpassQueryBuilder(area=areaId, elementType='way',
selector='"building" = "church"', out='center')
# query = overpassQueryBuilder(area=areaId, elementType='way',
selector='"building" = "train_station"', out='center')

query = overpassQueryBuilder(area=areaId, elementType='node', selector='"{}"
= "{}"'.format(searchSelector1, searchSelector2), out='center')

result = overpass.query(query)
print("Nr of entries: {}".format(result.countElements()))

coordinates = []
names = []
for entry in result.elements():
    if entry.tag('name'):
        names.append(entry.tag('name'))
    else:
        names.append("NaN")
    coordinates.append([entry.lat(), entry.lon()])
# print("Their names: \\n{}".format(names))
# print("Their coordinates: \\n{}".format(coordinates))
# #
for i in range(len(coordinates)):
    print("Name: ", names[i], " Coordinates: ", coordinates[i])

# insert data into numpy array
numpyarray = np.array(coordinates)
# print(numpyarray)
# print("\\n")
# plt.scatter(numpyarray[:, 0], numpyarray[:, 1], c =
range(len(coordinates)), alpha = 0.6)
# plt.show()

# create the pandas dataframe from numpy array
df = pd.DataFrame(numpyarray)
df.columns = ['lon', 'lat']
```

```
print(df)
print('\\n')

# Deleting the rows whose name is NaN:
# finding their index
namesWithoutNaN = []
for i in range(len(names)):
    # if names[i] == 'NaN':
    #     df.drop(i, axis = 0, inplace = True)
    # else:
    namesWithoutNaN.append(names[i])
print(len(namesWithoutNaN))

# # Reset the index so that the dropped rows are subsituted
df.reset_index(drop = True, inplace = True)
# print(df)
#
plt.scatter(df['lat'], df['lon'], c = range(len(df)), alpha = 0.6)
for i in range(len(namesWithoutNaN)):
    plt.annotate(namesWithoutNaN[i], xy = (df['lat'][i], df['lon'][i]), size
= 5)
#df.to_csv(r'churches_without_nan.csv', index=True, header=True)
df.to_csv(r'{}.csv'.format(searchSelector2), index=True, header=True)
plt.show()
```

Figure 9: Code of the OpenStreetMap Communicator program

10.2 contamination_GA1_0 (Processing)

```
import java.io.File;

GeneticOptimizer gO;

int resolution = 3;
StringList fieldNames;
ArrayList<Table> fieldTables;

int margin = 50;

void setup() {
  size(800,800);

  // Load data from OSM into an Array of Tables and Names - These will be
used throughout the program
  java.io.File folder = new java.io.File(dataPath("/Users/libar/TU
Wien/Master/GeneticAlgorithms/contaminationGA1_0/data/optimum"));
  String[] fileNames = folder.list();
  fieldNames = new StringList();
  fieldTables = new ArrayList<Table>();
  for (String fileName : fileNames) {
    // In case there are other files in the data folder we check fot the
right extension
    if (fileName.endsWith(".csv")) {
      fieldTables.add(loadTable("optimum/"+fileName, "header"));
      fieldNames.append(fileName);

    }
  }

  //Create the GeneticOptimizer, which also creates the Optimum from the CSV
Data
```

```
  gO = new GeneticOptimizer(fieldTables, margin, fieldNames);
  gO.initialize(1000);
  while (gO.generationCounter < 20) {
    gO.evolve();
  }
  gO.saveFittestAsTable(gO.population);
  gO.saveContaminatedAsTable(gO.contaminatedPopulation);

  println("DONE in " + millis()/1000.0 + " seconds");
  exit();
}

class GeneticOptimizer {
  ArrayList<FieldMap> population;
  ArrayList<FieldMap> contaminatedPopulation;
  IntList matingPool;
  StringList fieldNames;
  float margin;
  float minLon, maxLon, minLat, maxLat;
  float minX, maxX, minY, maxY;
  //float mutationRate = 0.2;
  float contaminationRate = 0.1;
  int optimumAverageNodes;
  int optimalNrOfPoints;
  float[] optimalPointRatio;
  float[] optimalConnectionRatio;
  float[] optimalNrOfConnections;
  FieldMap optimum;
  FieldMap fittestSolution;

  float maxFitness = 0.0;
  float currentFitness = 0.0;
  float averagePopFitness = 0.0;
  ArrayList<Float> averageFitnessArchive;
  ArrayList<Float> highestFitnessArchive;
  ArrayList<FieldMap> lowestFitnessMaps;
  //The following variables designate the min and max of possible mutation
rates
  //float mMax = 0.3;
  //float mMin = 0.1;
  //float mMax = 0.3;
  //float mMin = 0.03;
  float mMax = 0.5;
  float mMin = 0.1;
  int generationCounter = 0;

  GeneticOptimizer(ArrayList<Table> tables_, float margin_, StringList
fieldNames_) {
    population = new ArrayList<FieldMap>();
    contaminatedPopulation = new ArrayList<FieldMap>();
    matingPool = new IntList();
    margin = margin_;
    fieldNames = fieldNames_;
    optimalPointRatio = new float[fieldNames.size()];
    optimalNrOfConnections = new float[fieldNames.size()];
    optimalConnectionRatio = new float[fieldNames.size()];
    minX = margin;
    maxX = width-margin;
    minY = margin;
    maxY = height-margin;
    averageFitnessArchive = new ArrayList<Float>();
    highestFitnessArchive = new ArrayList<Float>();
    //An archive of the population0s lowest performing members for each
generation
    lowestFitnessMaps = new ArrayList<FieldMap>();

    //Create the optimum and transform the coordinates to pixel location
    minMaxCoordsTransform(tables_);
    optimum = new FieldMap(csvToCoords(tables_), margin, fieldNames);

    // Calculate an average amount of nodes that will be used in the
creation of new FieldMaps
    // and the optimum's ratios and numbers for later fitness calculation
    for (Table table : tables_) {
      optimalNrOfPoints += table.getRowCount();
    }
    optimumAverageNodes = optimalNrOfPoints / fieldNames.size() * 2;

    for (int i = 0; i < optimum.fieldArray.size(); i++) {
      optimalPointRatio[i] =
calcCloudRatio2(optimum.fieldArray.get(i).cloud);
      optimalNrOfConnections[i] = optimum.fieldArray.get(i).nrOfNeighbors;
      optimalConnectionRatio[i] =
calcConnectionRatio(optimum.fieldArray.get(i));
      println("Optimal PointRatio for " + fieldNames.get(i) + ": " +
optimalPointRatio[i]);
      println("Optimal ConnectionRatio for " + fieldNames.get(i) + ": " +
optimalConnectionRatio[i]);
      // DEPRECATED println("Optimal ConnectionRatio for " +
fieldNames.get(i) + ": " + optimalConnectionRatio[i]);
    }
    println("Optimal Nr of total Connections: " + optimalNrOfConnections);
    println("");
  }

  void initialize(int populationSize) {

    //Create as many FieldMaps as the population is big
    for (int i = 0; i < populationSize; i++) {
      ArrayList<ArrayList> coordinates = new ArrayList<ArrayList>();
      //Create in every FieldMap as many fields as there are FieldNames
      for (int j = 0; j < fieldNames.size(); j++) {
        //Create a random amount of Nodes for every field
        ArrayList<float[]> coordPairs = new ArrayList<float[]>();
        //Method that takes inspiration from the actual number of the
respective field
        int nrOfPairs = int(random(1,
optimum.fieldArray.get(j).nodeArray.size() * 2));
        //println("Nr of Pairs: " + nrOfPairs);
        for (int c = 0; c < nrOfPairs; c++) {
          //The following line is the standard random method
          float[] pair = {random(minX, maxX), random(minY, maxY)};
          //Now follows a new method that uses normal distribution which
makes sense as most
          //infrastructure is more prevalent close to the city centre
          //float xCoord = map(constrain(randomGaussian(), -1, 1), -1, 1,
minX, maxX);
          //float yCoord = map(constrain(randomGaussian(), -1, 1), -1, 1,
minY, maxY);
          //float[] pair = {xCoord, yCoord};
          coordPairs.add(pair);
        }
        coordinates.add(coordPairs);
      }
      population.add(new FieldMap(coordinates, margin, fieldNames));
    }
    //Initalize the fittest solution by setting it equal to the first member
of the population
    fittestSolution = population.get(0);

    println("Population Initialisation done in " + millis()/1000.0 + "
seconds");
  }

  void evolve() {
    calculateFitness();
    contamination();
    selection();
    reproduction();
  }

  void calculateFitness() {

    averagePopFitness = 0.0;
    // First the fitness for every field is calculated in terms of its
PointCloudRatio and then ConnectionNr
    // The data is stored
    for (FieldMap fieldMap : population) {
      //float maxConnectionDifference = 0;
      //Had to add the clear() instruction because of an error that added
Infitiy values to the array; probably from
      //the generagion before. didn't really solve it however.
      fieldMap.fieldPointFitness.clear();
      fieldMap.fieldConnectionFitness.clear();
      fieldMap.nodeNrFitness.clear();
      for (int i = 0; i < fieldMap.fieldArray.size(); i ++) {

        ////////////////////////////PointFitness///////////////////////////
        float pointRatio =
calcCloudRatio2(fieldMap.fieldArray.get(i).cloud);
        //println("Points of field: " + pointRatio + " Points of Opti: " +
optimalPointRatio[i]);
        //println("Field " + fieldMap.fieldArray.get(i).fieldName + " Ratio:
" + fieldMap.fieldArray.get(i).pointRatio);
        //println("Difference in ratio: " +
abs(fieldMap.fieldArray.get(i).pointRatio - optimalPointRatio[i]));
        //println(optimalPointRatio[i] + " - " + pointRatio + " = " +
abs(optimalPointRatio[i] - pointRatio) + " of available: " +
optimalPointRatio[i] + " | Mapped : " + map(abs(optimalPointRatio[i] -
pointRatio), 0, optimalPointRatio[i], 1,0));

fieldMap.fieldPointFitness.add(constrain(pow(map(abs(optimalPointRatio[i] -
pointRatio), 0, optimalPointRatio[i], 1, 0), 4), 0, 1));


////////////////////////////ConnectionFitness///////////////////////
        float connectionRatio =
calcConnectionRatio(fieldMap.fieldArray.get(i));
        fieldMap.fieldConnectionFitness.add(pow(map(abs(connectionRatio -
optimalConnectionRatio[i]), 1, 0, 0, 1), 4));
        //println("Single Field Connection Fitness: " +
fieldMap.fieldConnectionFitness.get(i));

        ////////////////////////////Node Number
Fitness///////////////////////
        float nodeDifference =
calcNodeDifference(fieldMap.fieldArray.get(i), optimum.fieldArray.get(i));
```

```
        fieldMap.nodeNrFitness.add(pow(map(nodeDifference, 0,
optimum.fieldArray.get(i).nodeArray.size(), 1, 0), 2));
      }

      float combinedPointFitness = 0;
      float combinedConnectionFitness = 0;
      float combinedNodeNrFitness = 0;
      int nrOfFields = fieldMap.fieldArray.size();
      for (int i = 0; i < nrOfFields; i ++) {
        combinedPointFitness += fieldMap.fieldPointFitness.get(i);
        //combinedConnectionFitness +=
fieldMap.fieldConnectionFitness.get(i);
        //combinedNodeNrFitness += fieldMap.nodeNrFitness.get(i);
      }
      combinedPointFitness = combinedPointFitness / nrOfFields;
      combinedConnectionFitness = constrain(combinedConnectionFitness /
nrOfFields, 0.0001, 1.0);
      combinedNodeNrFitness = combinedNodeNrFitness / nrOfFields;
      //println("");
      //println("CombinedPointFitness: " + combinedPointFitness);
      //println("CombinedConnectionFitness : " + combinedConnectionFitness);
      //println("CombinedNodeNumberFitness : " + combinedNodeNrFitness);
      fieldMap.overallFitness = combinedPointFitness;
      //fieldMap.overallFitness = (combinedPointFitness +
combinedConnectionFitness) / 2;
      //fieldMap.overallFitness = (combinedPointFitness +
combinedNodeNrFitness) / 2;
      //fieldMap.overallFitness = (combinedPointFitness +
combinedConnectionFitness + combinedNodeNrFitness + combinedNodeNrFitness +
combinedNodeNrFitness) / 5;
      //println("Overall Fitness : " + fieldMap.overallFitness);
      //println("");
      averagePopFitness += fieldMap.overallFitness;
    }
    averagePopFitness = averagePopFitness / population.size();
  }

  void selection() {
    generationCounter++;
    matingPool.clear();
    maxFitness = 0.0;
    for (FieldMap fieldMap : population) {
      if (fieldMap.overallFitness > maxFitness) {
        maxFitness = fieldMap.overallFitness;
      }
    }
    println("Generation " + generationCounter + ": ");
    highestFitnessArchive.add(maxFitness);
    println("Current Highest Fitness is: " + maxFitness + " and the highest
was " + currentFitness);
    averageFitnessArchive.add(averagePopFitness);
    println("The population's average fitness is " + averagePopFitness);
    println("Time spent to this point: " + millis()/1000.0/60 + " minutes");
    println("");
    for (int i = 0; i < population.size(); i++) {
      //float mappedFitness =
map(population.get(i).overallFitness,0,maxFitness,0,1);
      //mappedFitness * number -> linear fitness selection
      //int n = int(mappedFitness * 1000);  // Arbitrary multiplier, we can
also use monte carlo method
      // and pick two random numbers
      //Now follows an exponential  choice which rewards better fitness much
stronger
```

```
      float mappedFitness = map(population.get(i).overallFitness, 0,
maxFitness, 0, 6);
      int n = int(pow(mappedFitness, 3));
      for (int j = 0; j < n; j++) {
        matingPool.append(i);
      }
    }
    if (maxFitness > currentFitness) {
      currentFitness = maxFitness;
    }
  }

  void reproduction() {
    ArrayList<FieldMap> provisionalPopulation = new ArrayList<FieldMap>();
    for (int i = 0; i < population.size(); i++) {
      int a = int(random(matingPool.size()));
      int b = int(random(matingPool.size()));
      FieldMap partnerA = population.get(matingPool.get(a));
      FieldMap partnerB = population.get(matingPool.get(b));

      //Here begins the experimental Adaptive GA method
      FieldMap child;
      child = crossover(partnerA, partnerB);
      //child.mutate(calcMutationRate(child), minX, maxX, minY, maxY);
      //child.mutate2(calcMutationRate2(child), minX, maxX, minY, maxY);
      //child.mutate3(calcMutationRate(child), minX, maxX, minY, maxY,
optimum);
      child.mutate4(calcMutationRate(child), minX, maxX, minY, maxY,
optimum);
      //Here it ends

      provisionalPopulation.add(child);
    }
    population.clear();
    population = provisionalPopulation;
  }

  void contamination() {
    //Introducing the lowest performing member of the population into an
archive of lowperformers
    //Noting the index of the highest performing member. This is all done
with the following method:
    int fittestIndex = findLowestHighestFitness();

    FieldMap contaminated =
contaminationCrossover(population.get(fittestIndex),
lowestFitnessMaps.get(int(random(lowestFitnessMaps.size()))));
    contaminatedPopulation.add(contaminated);

  }

  FieldMap crossover(FieldMap partner1, FieldMap partner2) {
    ArrayList<ArrayList> childCoordinates = new ArrayList<ArrayList>();

    //println(partner1.coordinates.get(0).get(0)[1]);
    for (int i = 0; i < partner1.coordinates.size(); i ++) {
      ArrayList<float[]> childFieldCoords = new ArrayList<float[]>();
      ArrayList<float[]> partner1FieldCoords = partner1.coordinates.get(i);
      ArrayList<float[]> partner2FieldCoords = partner2.coordinates.get(i);

      for (int j = 0; j < partner1FieldCoords.size()/2; j++) {
        childFieldCoords.add(partner1FieldCoords.get(j));
      }
      for (int j = partner2FieldCoords.size()/2; j <
partner2FieldCoords.size(); j++) {
        childFieldCoords.add(partner2FieldCoords.get(j));
      }
      childCoordinates.add(childFieldCoords);
    }

    return new FieldMap(childCoordinates, margin, fieldNames);
  }

  FieldMap contaminationCrossover(FieldMap clean, FieldMap contaminated) {
    ArrayList<ArrayList> childCoordinates = new ArrayList<ArrayList>();

    //println(partner1.coordinates.get(0).get(0)[1]);
    for (int i = 0; i < clean.coordinates.size(); i ++) {
      ArrayList<float[]> childFieldCoords = new ArrayList<float[]>();
      ArrayList<float[]> cleanFieldCoords = clean.coordinates.get(i);
      ArrayList<float[]> contaminatedFieldCoords =
contaminated.coordinates.get(i);

      for (int j = 0; j < cleanFieldCoords.size(); j++) {
        childFieldCoords.add(cleanFieldCoords.get(j));
      }

      for (int j = 0; j < cleanFieldCoords.size(); j++) {
        if (random(1) < contaminationRate) {
          childFieldCoords.set(j,
contaminatedFieldCoords.get(int(random(contaminatedFieldCoords.size()))));
        }
      }


      childCoordinates.add(childFieldCoords);
    }

    return new FieldMap(childCoordinates, margin, fieldNames);
  }

  ///////////////Helper Methods///////////////

  int findLowestHighestFitness() {
    int fittestIndex = 0;
    float highestGenerationalFitness = 0.0;
    FieldMap lowestFitnessMap = population.get(0);
    for (int i = 0; i < population.size(); i++) {
      if (population.get(i).overallFitness > highestGenerationalFitness) {
        fittestIndex = i;
        highestGenerationalFitness = population.get(i).overallFitness;
      }
      if (lowestFitnessMap.overallFitness <
population.get(i).overallFitness) {
        lowestFitnessMap = population.get(i);
      }
    }
    lowestFitnessMaps.add(lowestFitnessMap);
    return fittestIndex;
  }

  //Method used for the optimum that turns csv table coordinates into an
ArrayList
  //of an ArrayList of Float[] (yes, it's a nested ArrayList!)
  ArrayList<ArrayList> csvToCoords(ArrayList<Table> tables) {
    ArrayList<ArrayList> fields = new ArrayList<ArrayList>();
    //Iteratate through every table and create an ArrayList of the
respective field's coords
    for (int i = 0; i < tables.size(); i ++) {
      ArrayList<float[]> coordinates = new ArrayList<float[]>();
      for (TableRow row : tables.get(i).rows()) {
        //Here the coords are taken from csv and then mapped via the
established min and max values
        float[] coords = {map(row.getFloat("lon"), minLon, maxLon, minX,
maxX), map(row.getFloat("lat"), minLat, maxLat, minY, maxY)};
        coordinates.add(coords);
      }
      fields.add(coordinates);
    }
    println("Min and Max Coordinates: " + minLon + " " + maxLon + " " +
minLat + " " + maxLat);
    println("Min and Max Pixels: " + minX + " " + maxX + " " + minY + " " +
maxY);
    return fields;
  }

  void minMaxCoordsTransform(ArrayList<Table> tables) {
    ArrayList<float[]> coordinates = new ArrayList<float[]>();
    for (Table table : tables) {
      for (TableRow row : table.rows()) {
        float[] coords = {row.getFloat("lon"), row.getFloat("lat")};
        coordinates.add(coords);
      }
    }
    float[] lons = new float[coordinates.size()];
    float[] lats = new float[coordinates.size()];
    for (int j = 0; j < coordinates.size(); j++) {
      lons[j] = coordinates.get(j)[0];
      lats[j] = coordinates.get(j)[1];
    }
    minLon = min(lons);
    maxLon = max(lons);
    minLat = min(lats);
    maxLat = max(lats);
    minX = map(minLon, minLon, maxLon, margin, width-margin);
    maxX = map(maxLon, minLon, maxLon, margin, width-margin);
    minY = map(minLat, minLat, maxLat, margin, height-margin);
    maxY = map(maxLat, minLat, maxLat, margin, height-margin);
  }

  //Method for Fitness Clac that evaluates the ration of points occupied in
pointcloud
  float calcCloudRatio(PointCloud pointCloud) {
    float size = pointCloud.cloud.length * pointCloud.cloud[0].length;
    float occupied = 0;
    for (int i = 0; i < pointCloud.cloud.length; i++) {
      for (int j = 0; j < pointCloud.cloud[i].length; j++) {
        if (pointCloud.cloud[i][j] == 1) {
          occupied++;
        }
      }
    }
    //println("Ratio: " + occupied/size + " Occpuied: " + occupied + " of "
+ size + " points available");
    return occupied/size;
  }

  float calcCloudRatio2(PointCloud pointCloud) {
    float occupied = 0;
    for (int i = 0; i < pointCloud.cloud.length; i++) {
      for (int j = 0; j < pointCloud.cloud[i].length; j++) {
        if (pointCloud.cloud[i][j] == 1) {
          occupied++;
        }
      }
    }
    //println("Ratio: " + occupied/size + " Occpuied: " + occupied + " of "
+ size + " points available");
    return occupied;
  }

  float availablePoints(PointCloud pointCloud) {
    return pointCloud.cloud.length * pointCloud.cloud[0].length;
  }

  float calcConnectionRatio(Field field) {
    float nrOfNodes = field.nodeArray.size();
    float nrOfConnections = 0.0;
    for (Node node : field.nodeArray) {
      if (node.neighbors.size() > 0) {
        nrOfConnections++;
      }
    }
    return nrOfConnections/nrOfNodes;
  }

  float calcNodeDifference(Field field, Field optimumField) {
    float nrOfNodes = field.nodeArray.size();
    float optimumNrOfNodes = optimumField.nodeArray.size();
    float diff = constrain(abs(optimumNrOfNodes - nrOfNodes), 0,
optimumNrOfNodes);
    //println("DIFFERENCE: " + diff + " (Opti: " + optimumNrOfNodes + "
Field: " + nrOfNodes);
    return diff;
  }

  float minMaxNormalize(float value, float min, float max) {
    return (value-min) / (max-min) + 0.0;
  }

  //method from the paper on electric vehicles
  float calcMutationRate(FieldMap child) {
    float childFitness = calcSingleFitness(child);
    float mutationRate = mMax;
    float error = 1.0 - averagePopFitness;
    if (abs(maxFitness - childFitness) <= error) {
      mutationRate = (mMin + mMax) / 2;
    } else if (childFitness > averagePopFitness + error) {
      mutationRate = mMax - ((mMax - mMin) * (childFitness -
averagePopFitness)) / (maxFitness - averagePopFitness);
    } else if (childFitness < averagePopFitness - error) {
      mutationRate = mMax;
    }

    return constrain(mutationRate, mMin, mMax);
  }

  //own method
  float calcMutationRate2(FieldMap child) {
    float childFitness = calcSingleFitness(child);
    float mutationRate;
    if (childFitness < averagePopFitness) {
      mutationRate = mMax;
    } else {
      mutationRate = mMin;
    }
    if (averageFitnessArchive.size() > 0) {
      float lastAvgFitness =
averageFitnessArchive.get(averageFitnessArchive.size()-1);
      if (abs(maxFitness - lastAvgFitness) < 0.1) {
        mutationRate = mMax + 0.2;
      }
    }

    return constrain(mutationRate, mMin, mMax);
  }

  //This function is called to calc the fitness of child that is to be
mutated to be able to
  //to calculate the mutationrate of the child
  float calcSingleFitness (FieldMap child) {
    for (int i = 0; i < child.fieldArray.size(); i ++) {
      child.fieldArray.get(i).pointRatio =
calcCloudRatio2(child.fieldArray.get(i).cloud) + 0.00000001;
      child.fieldPointFitness.add(constrain(pow(map(abs(optimalPointRatio[i]
- child.fieldArray.get(i).pointRatio), 0, optimalPointRatio[i], 1, 0), 4),
0, 1));

      float connectionRatio = calcConnectionRatio(child.fieldArray.get(i));
      child.fieldConnectionFitness.add(pow(map(abs(connectionRatio -
optimalConnectionRatio[i]), 1, 0, 0, 1), 200));
    }

    float combinedPointFitness = 0;
    float combinedConnectionFitness = 0;
    int nrOfFields = child.fieldPointFitness.size();
    for (int i = 0; i < nrOfFields; i ++) {
      combinedPointFitness += child.fieldPointFitness.get(i);
      combinedConnectionFitness += child.fieldConnectionFitness.get(i);
    }
    combinedPointFitness = combinedPointFitness / nrOfFields;
    combinedConnectionFitness = combinedConnectionFitness / nrOfFields;
    //return (combinedPointFitness + combinedConnectionFitness) / 2;
    return combinedPointFitness;
  }

/////////Output Methods/////////

  void saveFittestAsTable(ArrayList<FieldMap> population) {
    FieldMap fittestPhenotype = fittestPheno(population);
    for (Field field : fittestPhenotype.fieldArray) {
      Table table = new Table();
      table.addColumn("");
      table.addColumn("lon");
      table.addColumn("lat");
      for (int i = 0; i < field.nodeArray.size(); i++) {
        float[] coords = coordsBackTransform(field.nodeArray.get(i).loc);
        TableRow newRow = table.addRow();
        newRow.setInt("", i);
        newRow.setFloat("lon", coords[0]);
        newRow.setFloat("lat", coords[1]);
      }
      saveTable(table, "fittestOutput/" + field.fieldName + ".csv");
    }
    String[] analysis = new String[3];
    analysis[0] = ("Current Highest Fitness is: " + maxFitness + " and the 
highest was " + currentFitness + "\\n");
    analysis[1] = ("The population's average fitness is " + 
averagePopFitness + "\\n");
    analysis[2] = ("Pop of " + population.size() + " in " + millis()/1000.0 
+ " seconds");
    saveStrings("fittestOutput/analysis.txt", analysis);
  }

  void saveContaminatedAsTable(ArrayList<FieldMap> contaminatedPopulation) {
    //FieldMap fittestPhenotype = fittestPheno(contaminatedPopulation);
    FieldMap fittestPhenotype = 
contaminatedPopulation.get(contaminatedPopulation.size()-1);
    for (Field field : fittestPhenotype.fieldArray) {
      Table table = new Table();
      table.addColumn("");
      table.addColumn("lon");
      table.addColumn("lat");
      for (int i = 0; i < field.nodeArray.size(); i++) {
        float[] coords = coordsBackTransform(field.nodeArray.get(i).loc);
        TableRow newRow = table.addRow();
        newRow.setInt("", i);
        newRow.setFloat("lon", coords[0]);
        newRow.setFloat("lat", coords[1]);
      }
      saveTable(table, "contaminatedOutput/" + field.fieldName + ".csv");
    }
    //String[] analysis = new String[3];
    //analysis[0] = ("Current Highest Fitness is: " + maxFitness + " and the 
highest was " + currentFitness + "\\n");
    //analysis[1] = ("The population's average fitness is " + 
averagePopFitness + "\\n");
    //analysis[2] = ("Pop of " + population.size() + " in " + 
millis()/1000.0 + " seconds");
    //saveStrings("contaminatedOutput/analysis.txt", analysis);
  }

  FieldMap fittestPheno(ArrayList<FieldMap> population) {
    float highestFitness = 0;
    FieldMap fittestPheno = population.get(0);
    for (FieldMap fieldMap : population) {
      if (fieldMap.overallFitness >= highestFitness) {
        fittestPheno = fieldMap;
        highestFitness = fieldMap.overallFitness;
      }
    }
    return fittestPheno;
  }

  float[] coordsBackTransform(PVector loc) {
    float lon = map(loc.y, minY, maxY, minLon, maxLon);
    float lat = map(loc.x, minX, maxX, minLat, maxLat);
    float[] coords = {lon, lat};
    return coords;
  }
}
```

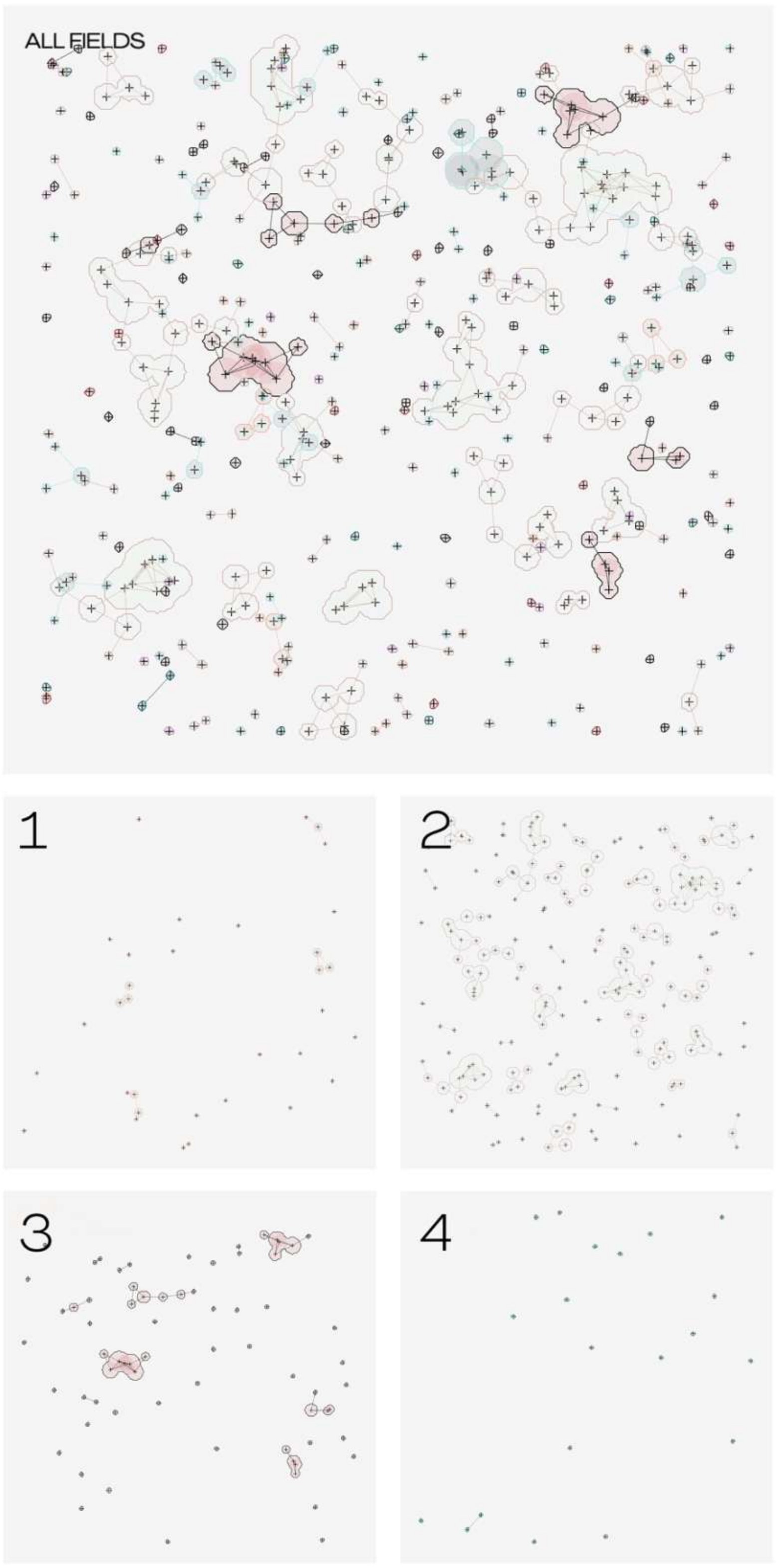
Figure 10: Code of the genetic algorithm with integrated contamination function

10.3 fieldmapVisualiser1_0 (Processing)

```
import java.io.File;

FieldMap fieldMap;
float margin;

PImage map;
float mapX, mapY;

ArrayList<Table> fields;
int fieldsShown = 0;

// The booleans that determine what is shown
PFont agrandirTitle, agrandirBody;
int textSize = 25;
boolean frameRateControl = false;
boolean showSingleField = true;
boolean showConnections = true;

void setup() {
  size(900, 900, P3D);
  pixelDensity(displayDensity());
  cursor(ARROW);
  frameRate(60);
  smooth(8);
  background(244, 245, 244);
  rectMode(CENTER);
  ellipseMode(CENTER);

  //agrandir = loadFont("Helvetica-Oblique-25.vlw");
  agrandirTitle = loadFont("AgrandirVariable-Regular_Wide-25.vlw");
  agrandirBody = loadFont("AgrandirVariable-Regular_Wide-Light-18.vlw");
  //textMode(SHAPE);
  margin = width / 18;
  //Load all the csv filnes into an ArrayList which we will pass into
fieldMap
  java.io.File folder = new java.io.File(dataPath

("/directory/JAVA_field_condition_visualiser_MULTIPLE_2/data/run06/conta"));
  String[] filenames = folder.list();
  //Create the fieldNames Array that will hold the names of the field
  StringList fieldNames = new StringList();
  printArray(filenames);
  fields = new ArrayList<Table>();
  for (String filename : filenames) {
    // In caseb there are other files in the data folder we check fot the
right extension
    if (filename.endsWith(".csv")) {
      fields.add(loadTable("run06/conta/" + filename, "header"));
      println(filename);
      fieldNames.append(filename);
    }
  }
  println(fieldNames);

  fieldMap = new FieldMap(fields, margin, fieldNames);
  fieldMap.pickDisplay(fieldsShown, showSingleField, showConnections);

}

void draw() {
  background(244, 245, 244);
  fieldMap.pickDisplay(fieldsShown, showSingleField, showConnections);
  if (frameRateControl) {
    println(frameRate);
  }

}

void keyPressed() {
  if (key == 'a' || key == 'A') {
    showSingleField = !showSingleField;
  }
  if (key == 'c' || key == 'C') {
    showConnections = !showConnections;
  }
  if (key == 'f' || key == 'F') {
    frameRateControl = !frameRateControl;
  }
  if (key == 'n' || key == 'N') {
    if (fieldsShown < fields.size()-1) {
      fieldsShown ++;
    }
  }
  if (key == 'm'|| key == 'M') {
    if (fieldsShown > 0) {
      fieldsShown --;
    }
  }
  if (key == 'p'|| key == 'P') {
    saveFrame("output/run06/conta/conta_" +
fieldMap.fieldArray.get(fieldsShown).fieldName + ".jpg");
  }
}

class FieldMap {
  ArrayList<Field> fieldArray;
  ArrayList<float[]> coordinates;
  ArrayList<color[]> colorArray;

  float minLon_, maxLon_, minLat_, maxLat_;

  // margin determines how much space is to be left to sketch border
  FieldMap(ArrayList<Table> csvTables, float margin, StringList fieldNames)
{

    // Initiate the fieldArray in which all fields will find a place
    fieldArray = new ArrayList<Field>();

    colorArray = new ArrayList<color[]>();
    for (color[] colorDuo : colorSet) {
      colorArray.add(colorDuo);
    }
    addAdditionalColors();
    println("ColorArraySize: " + colorArray.size());

    // The main part where all the coordinates are passed down and the
repective fields are created
    for (int i = 0; i < csvTables.size(); i ++) {

      ArrayList<float[]> coordinates = new ArrayList<float[]>();
      // Load the CSV Data into an ArrayList of Float[] that each contains
one location
      ArrayList<float[]> provisionalCoordinates = new ArrayList<float[]>();
      for (TableRow row : csvTables.get(i).rows()) {
        float[] coords = {row.getFloat("lon"), row.getFloat("lat")};
        provisionalCoordinates.add (coords);

      }

      //Get the max and min numbers in Coordinats to convert to Sketch size
before passing them on to nodes
      float minLon, maxLon, minLat, maxLat;
      float[] lons = new float[provisionalCoordinates.size()];
      float[] lats = new float[provisionalCoordinates.size()];
      for (int j = 0; j <provisionalCoordinates.size(); j++) {

        lons[j] = provisionalCoordinates.get(j)[0];
        lats[j] = provisionalCoordinates.get(j)[1];

      }
      minLon = min(lons); maxLon = max(lons); minLat = min(lats); maxLat =
max(lats);
      minLon_ = minLon; maxLon_ = maxLon; minLat_ = minLat; maxLat_ =
maxLat;
      for (float[] coord : provisionalCoordinates) {

        float[] mappedCoords = {map(coord[0], minLon, maxLon, margin, width-
margin),
                                map(coord[1], minLat, maxLat, margin,
height-margin)};
        coordinates.add(mappedCoords);

      }
      //There needs to be a variable that controls the size of circles
dynamically
      //otherwse, if there are too many entries, the circle becomes too big
      float neighborDistance = margin;
      // Finally, add every field to the array of fields
      fieldArray.add(new Field(coordinates, neighborDistance,
fieldNames.get(i), colorArray.get(i)));
    }
  }

  void addNode(int x, int y) {
    for (Field field : fieldArray) {
      field.addNode(x, y);
    }
  }

  void pickDisplay(int index, boolean single, boolean connections) {
    if (single) {
      fieldArray.get(index).display();
      if (connections) {
        fieldArray.get(index).displayConnections();
      }
      fieldArray.get(index).showBoundaries();
      fill(0);
      textFont(agrandirTitle);
      text(fieldArray.get(index).fieldName.toUpperCase(), textSize,
textSize*2);
      textFont(agrandirBody);
      text("\\nNumber of Nodes: " + fieldArray.get(index).nodeArray.size(),
textSize, textSize*2);
      text("\\n\\nNumber of Connections: " +
fieldArray.get(index).nrOfNeighbors, textSize, textSize*2);
    } else {
      for (Field field : fieldArray) {
        field.display();
        if (connections) {
          field.displayConnections();
        }
        field.showBoundaries();
      }
      fill(0);
      textFont(agrandirTitle);
      text("ALL FIELDS", textSize, textSize*2);
    }
  }

  void displayText() {
    // Insert Text
    fill(0);
    textFont(agrandirTitle);
    text("CHURCHES", textSize, textSize*2);
    textFont(agrandirBody);
  }

  void displayAll() {
    for (Field field : fieldArray) {
      field.display();
    }
  }

  void displayConnections() {
    for (Field field : fieldArray) {
      field.displayConnections();
    }
  }

  void showPoints() {
    for (Field field : fieldArray) {
      field.showPoints();
    }
  }

  void showBoundaries() {
    for (Field field : fieldArray) {
      field.showBoundaries();
    }
  }

  void checkIntersections() {
    for (Field field : fieldArray) {
      field.checkIntersections();
    }
  }

  void printy() {
    for (Field field : fieldArray) {
      field.printy();
    }
  }
  void addAdditionalColors() {
    for (int i = 0; i < 20; i ++) {
      color[] colorDuo = {color(random(255),random(255),random(255),20),
color(random(255),random(255),random(255))};
      colorArray.add(colorDuo);
    }
  }
  // First color is Fill, second Stroke
  // These Colors have been handpicked, but 20 additional random ones will
be added during initialisation
  color[][] colorSet = {{color(35, 87, 137, 20), color(193, 41, 46)},
                        {color(45, 45, 42, 20), color(15, 113, 115)},
                        {color(194, 1, 20, 20), color(12, 18, 12)},
                        {color(101, 138, 86, 20), color(249, 173, 160)},
                        {color(224, 242, 233, 20), color(206, 181, 167)},
                        {color(55, 63, 71, 20), color(167, 226, 227)},
                        {color(184, 51, 106, 20), color(196, 144, 209)},
                        {color(76, 76, 71, 20), color(45, 45, 42)},
                        {color(178, 103, 94, 20), color(100, 69, 54)},
                        {color(227, 190, 195, 20), color(99, 107, 97)}};
}

class Field {
  PointCloud cloud;
  ArrayList<Node> nodeArray;
  String fieldName;
  float neighborDistance;
  float markerSize = 10;
  color[] colorSet;
  int nrOfNeighbors = 0;

  // neighborDistance designates the maximum distance to still be qualified
as a neighbor
  Field(ArrayList<float[]> coordinates, float neighborDistance_, String
fieldName_, color[] colorSet_) {

    // Store the field's passed down name and remove the file extension
    fieldName = fieldName_.substring(0, fieldName_.lastIndexOf('.'));

    //
    colorSet = colorSet_;

    // Create the point cloud that will check for the boundaries
    cloud = new PointCloud(3, colorSet[1]);

    // Create and fill the field's nodes with the passed down coordinates
    nodeArray = new ArrayList<Node>();
    for (float[] coordinate : coordinates) {
      nodeArray.add(new Node(coordinate[0], coordinate[1]));
    }
    neighborDistance = neighborDistance_;
    findNeighbors();
    calculateSize();
    cloud.checkIntersections(nodeArray);
  }

  // Calculate the number of neighbors for every node based on dist()
functions
  void findNeighbors() {
    for (Node node : nodeArray) {
      for (Node otherNode : nodeArray) {
        float distance = dist(node.loc.x, node.loc.y, otherNode.loc.x,
otherNode.loc.y);
        if (distance < neighborDistance && distance > 0.0) {
          node.neighbors.add(otherNode);
          if (!otherNode.neighborNames.contains(node.name)) {
            node.neighborNames.add(otherNode.name);
            nrOfNeighbors++;
          }
        }
      }
    }
  }

  // Calculate the size of every node based on their number of neighbors so
as to know how big their circle is
  void calculateSize() {
    for (Node node : nodeArray) {
      node.size = constrain(markerSize * node.neighbors.size(), markerSize,
markerSize*5); ///1.5;
    }
  }

  // addNode() is called upon mousePressed() and adds a Node to the field
  void addNode(int x, int y) {
    nodeArray.add(new Node(x, y));
    for (Node node : nodeArray) {
      node.neighbors.clear();
    }
    findNeighbors();
    calculateSize();
  }

/////////////////////////////////////////////////////////////////////

  void display() {
    for (Node node : nodeArray) {
      stroke(0);
      strokeWeight(1);
      noFill();
      //rect(node.loc.x, node.loc.y, markerSize, markerSize);
      marker(node.loc.x, node.loc.y, markerSize);
      if (node.size >= markerSize) {
        fill(colorSet[0]);
        noStroke();
        ellipse(node.loc.x, node.loc.y, node.size, node.size);
      }
    }
  }

  //Display the lines of every node with their neighbors
  void displayConnections() {
    for (Node node : nodeArray) {
      for (Node neighbor : node.neighbors) {
        strokeWeight(0.5);
        stroke(colorSet[1]);
        line(node.loc.x, node.loc.y, neighbor.loc.x, neighbor.loc.y);
      }
    }
  }

/////////////////////////////////////////////////////////////////////

  void showPoints() {
    cloud.showPoints();
  }

  void showBoundaries() {
    cloud.calcBoundaries();
  }

  void checkIntersections() {
    cloud.checkIntersections(nodeArray);
  }

  void marker(float x, float y, float diameter) {
    float radius = diameter /2;
    line(x-radius, y, x+radius, y);
    line(x, y-radius, x, y + radius);
  }
}

class Node {
  PVector loc;
  ArrayList<Node> neighbors;
  ArrayList<String> neighborNames;
  float size;
  String name;

  Node(float longitude, float latitude) {
    loc = new PVector(longitude, latitude);
    neighbors = new ArrayList<Node>();
    neighborNames = new ArrayList<String>();
    name = str(loc.x + loc.y);
  }

  void printy() {
    println("I have " + neighbors.size() + " neighbors and they are at: " );
    for (Node neighbor : neighbors) {
      println("Lon: " + neighbor.loc.x + " and Lat: " + neighbor.loc.y);
    }
    println(size);
    println("");
  }
}

class PointCloud {

  PVector a = new PVector(0,0);
  PVector b = new PVector(0,0);
  PVector c = new PVector(0,0);
  PVector d = new PVector(0,0);

  float[][] cloud;
  int cols, rows;
  int resolution;

  color boundaryColor;

  PointCloud(int resolution_, color boundaryColor_) {
    resolution = resolution_;
    cols = width/resolution + 1;
    rows = height/resolution + 1;
    cloud = new float[cols][rows];
    for (int i = 0; i < cols; i++) {
      for (int j = 0; j < rows; j++) {
        cloud[i][j] = 0;
      }
    }
    boundaryColor = boundaryColor_;
  }

  void checkIntersections(ArrayList<Node> NodeArray) {
    for (Node node : NodeArray) {
      if (node.size > 0) {
        for (int i = 0; i < cols; i++) {
          for (int j = 0; j < rows; j++) {
            if (dist(i*resolution, j*resolution, node.loc.x, node.loc.y) <=
node.size/2) {
              cloud[i][j] = 1;
            }
          }
        }
      }
    }
  }

  // Shows the Points in the grid for control purposes
  void showPoints() {
    //for (int i = 0; i < cols; i++) {
    //  for (int j = 0; j < rows; j++) {
    //    if (cloud[i][j] == 0) {
    //      stroke(0);
    //      strokeWeight(0.5);
    //      point(i*resolution, j*resolution);
    //    } else {
    //      stroke(193, 41, 46);
    //      strokeWeight(0.5);
    //      point(i*resolution, j*resolution);
    //    }
    //  }
    //}
  }

/////////////////////////////////////////////////////////////////////

  //draw the boundary line
  void calcBoundaries() {
    strokeWeight(1);
    stroke(boundaryColor);
    for (int i = 0; i < cols-1; i++) {
      for (int j = 0; j < rows-1; j++) {
        float x = i * resolution;
        float y = j * resolution;
        a = new PVector(x + resolution*0.5, y);
        b = new PVector(x + resolution, y+resolution*0.5);
        c = new PVector(x + resolution*0.5, y+resolution);
        d = new PVector(x, y+resolution*0.5);
        int state = getState(ceil(cloud[i][j]), ceil(cloud[i+1][j]),
                        ceil(cloud[i+1][j+1]), ceil(cloud[i][j+1]));
        switch (state) {
          case 1:
            boundLine(c,d);
            break;
          case 2:
            boundLine(b,c);
            break;
          case 3:
            boundLine(b,d);
            break;
          case 4:
            boundLine(a,b);
            break;
          case 5:
            boundLine(a,d);
            boundLine(b,c);
            break;
          case 6:
            boundLine(a,c);
            break;
          case 7:
            boundLine(a,d);
            break;
          case 8:
            boundLine(a,d);
            break;
          case 9:
            boundLine(a, c);
            break;
          case 10:
            boundLine(a,b);
            boundLine(c,d);
            break;
          case 11:
            boundLine(a,b);
            break;
          case 12:
            boundLine(b,d);
            break;
          case 13:
            boundLine(b,c);
            break;
          case 14:
            boundLine(c,d);
            break;
        }
      }
    }
  }

  //Used to calculate the state needed for the drawing of the boundary
  int getState(int a, int b, int c, int d) {
    return a * 8 + b * 4 + c * 2 + d;
  }

  //Used to draw boundaries in calcBoundaries()
  void boundLine(PVector v1, PVector v2) {
    line(v1.x, v1.y, v2.x, v2.y);
  }
}
```

Figure 11: Code of the fieldmap visualiser

11 Appendix B: Additional fieldmap



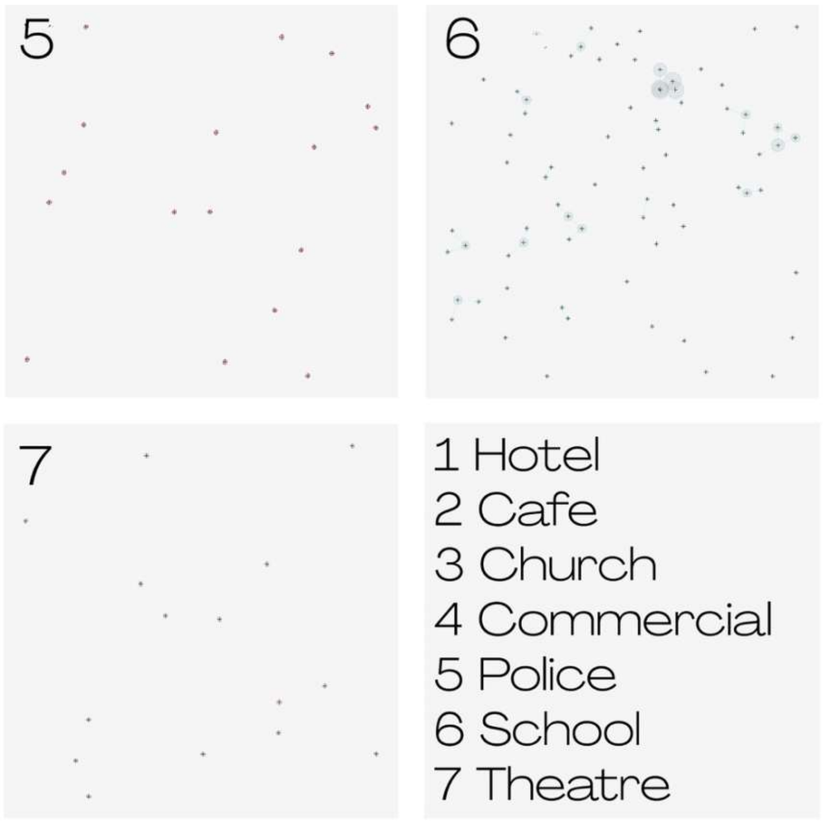
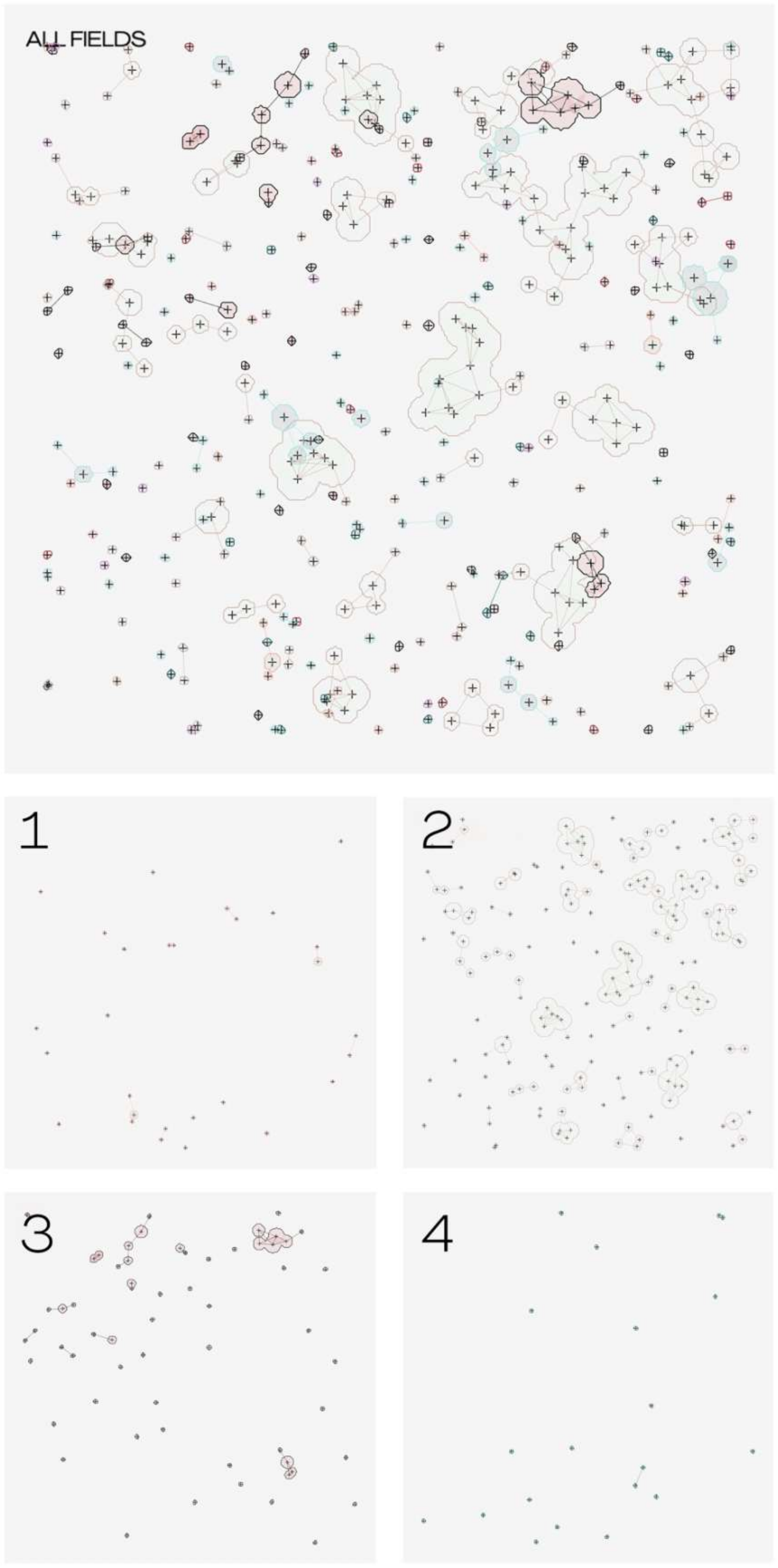


Figure 12: Run 6 uncontaminated fieldmap

Field Name	Nr. of Nodes	Nr. of Connections
Hotel	34	12
Cafe	205	244
Church	72	44
Commercial	21	1
Police	17	0
School	76	38
Theatre	16	0

Table 5: Numerical data of run 6 uncontaminated fieldmap


ALL FIELDS
1
2
3
4

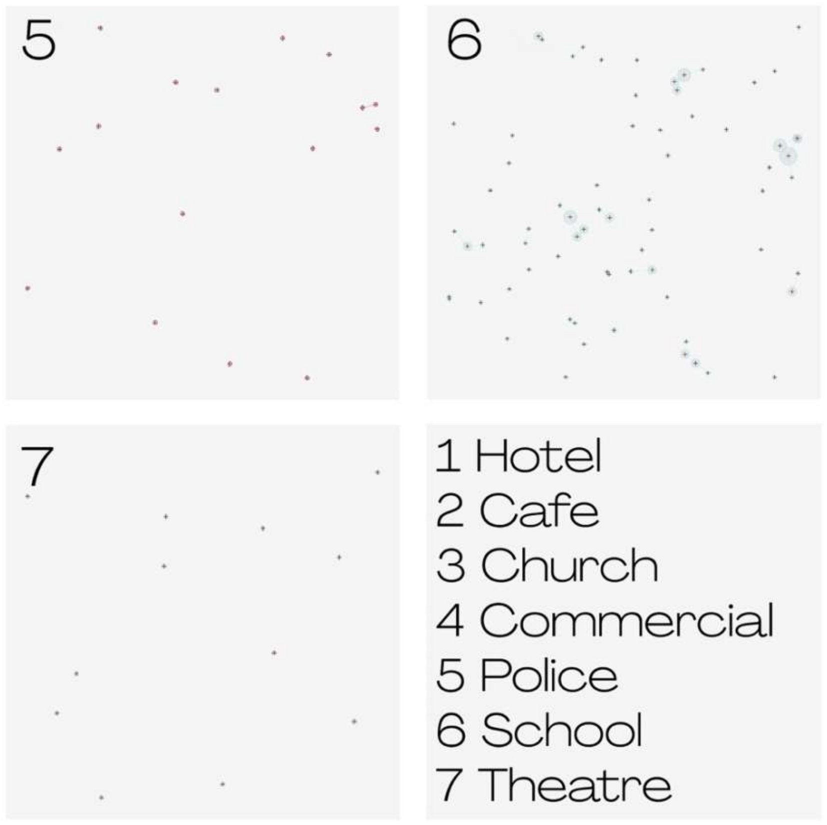


Figure 13: Run 6 contaminated fieldmap

Field Name	Nr. of Nodes	Nr. of Connections
Hotel	34	8
Cafe	205	246
Church	72	32
Commercial	22	2
Police	17	1
School	76	32
Theatre	16	0

Table 6: Numerical data of run 6 contaminated fieldmap

12 Appendix C List of Figures and Tables

List of Figures:

List of Tables: