POLITECHNIKA POZNAŃSKA

Agnieszka Mensfelt

The Application of Dissimilarity Measures for 3D Structures to Improve the Effectiveness of Evolutionary Design

Doctoral dissertation submitted in partial fulfilment

of the requirements for the degree of Doctor of Philosophy

Supervisor: Maciej Komosinski, Ph.D., Dr. Habil., Assoc. Prof.

Poznań, Poland
2023

Acknowledgments

I want to express my deep appreciation to my supervisor, Assoc. Prof. Maciej Komosinski, for his guidance during this research and all the work we have done together, which significantly contributed to my professional growth. Additionally, I am thankful to Prof. Roman Słowiński, the former head of the Intelligent Decision Support System Laboratory, and Prof. Jerzy Stefanowski, the current head of the Machine Learning Laboratory, for creating an excellent research environment and their support during my work there. I would also like to extend my thanks to my fellow Complex Systems Laboratory member, Konrad Miazga, for providing invaluable critical comments that helped me improve this thesis. Furthermore, I am grateful to my colleague, Dr. Habil. Robert Susmaga, who has been a constant source of professional and moral encouragement throughout this work. Last but not least, I want to express my gratitude to my family and friends. In particular, I am thankful to my ex-husband, Janusz, for his support and relentless motivation to pursue my goals.

Abstract

Evolutionary algorithms are a powerful tool for solving hard optimization problems, such as designing three-dimensional structures for various applications. Unlike human designers, these algorithms have the potential to explore an infinite solution space more extensively without requiring expert knowledge. However, the problem of evolutionary design, often characterized by a multimodal and rugged fitness landscape, still poses a challenge, and there is still room for improvement in optimization methods applied to it. This work explores the possibility of enhancing the effectiveness of evolutionary search by utilizing phenetic dissimilarity measures for 3D structures.

Developing a dissimilarity measure for 3D structures is a challenging problem in itself since there is no ground truth for the dissimilarity values. In this work, various dissimilarity measures are implemented and compared, including genotype-based, morphology graph-based, shape descriptor-based, and spatial distribution-based measures. Additionally, human perception of similarity is investigated as a potential reference point. The implemented measures are used to test the global convexity of the fitness landscape using the fitness-distance correlation (FDC) analysis in four optimization tasks, using three different genetic representations. In some of the cases examined, the fitness-distance correlation is negative, which may imply a global convexity of the fitness landscape.

With the aim to facilitate the effective search of the solution space, enhanced genetic operators are introduced, including Targeted Sequential Mutation (TSM), Distance Minimizing Crossover (DSX), Equidistance Minimizing Crossover (EMX), Distance Preserving Crossover (DPX), and Similarity-Based Crossover (SBX). The goal of these operators is to adjust the distance between parent and offspring solutions according to the selected dissimilarity measure. The performance of these operators is evaluated against the standard ones in a series of computational experiments. The TSM operator outperforms the standard mutation operator in most cases, whereas the proposed crossover operators achieve results comparable to the standard ones.

In the next strand of work, dissimilarity measures are applied in diversity maintenance techniques, which are methods for encouraging broader exploration of the solution space. The study compares the standard evolutionary algorithm with niching (local and global variants), novelty search (local and global variants), and the Non-dominated Sorting Genetic Algorithm II (NSGA-II), utilizing different dissimilarity measures. The analysis of the results reveals how different algorithms and dissimilarity measures influence the outcome of the evolution in terms of fitness and heterogeneity of the solutions.

Zastosowanie miar niepodobieństwa struktur 3D do zwiększenia efektywności projektowania ewolucyjnego

Streszczenie rozprawy w języku polskim

Algorytmy ewolucyjne są skutecznym narzędziem do rozwiązywania trudnych problemów optymalizacyjnych, takich jak projektowanie trójwymiarowych struktur dla różnorakich zastosowań. W przeciwieństwie do ludzkich projektantów, algorytmy te są w stanie szerzej eksplorować nieskończoną przestrzeń rozwiązań, nie wymagając przy tym wiedzy eksperckiej. Jednak problem projektowania ewolucyjnego, często charakteryzujący się multimodalnym i nieregularnym krajobrazem przystosowania, nadal stanowi wyzwanie i wciąż istnieje pole do poprawy stosowanych do niego metod optymalizacji. W niniejszej pracy zbadano możliwość zwiększenia skuteczności przeszukiwania ewolucyjnego poprzez wykorzystanie miar niepodobieństwa fenetycznego dla struktur 3D.

Opracowanie miary niepodobieństwa dla struktur 3D jest samo w sobie trudnym problemem, ponieważ nie jest znana obiektywna wartość niepodobieństwa. W niniejszej pracy zaimplementowano i porównano różne miary niepodobieństwa, w tym oparte na genotypie, grafie reprezentującym morfologię, deskryptorach kształtu i rozkładzie przestrzennym. Dodatkowo zbadano ludzkie postrzeganie podobieństwa jako potencjalny punkt odniesienia. Zaimplementowane miary są wykorzystywane do testowania globalnej wypukłości krajobrazu przystosowania przy użyciu analizy korelacji pomiędzy przystosowaniem a odległością do optimum (*fitness-distance correlaction*, FDC) w czterech zadaniach optymalizacyjnych, przy użyciu trzech różnych reprezentacji genetycznych. W niektórych z badanych przypadków korelacja wartości funkcji przystosowania i odległości jest ujemna, co może sugerować globalną wypukłość krajobrazu przystosowania.

W celu usprawnienia efektywności przeszukiwania przestrzeni rozwiązań, zaproponowano ulepszone operatory genetyczne, w tym Celowaną Mutację Sekwencyjną (*Targeted Sequential Mutation*, TSM), Krzyżowanie Minimalizujące Odległość (*Distance Minimizing Crossover*, DSX), Krzyżowanie Minimalizujące Równoodległość (*Equidistance Minimizing Crossover*, EMX), Krzyżowanie Zachowujące Odległość (*Distance Preserving Crossover*, DPX) i Krzyżowanie Bazujące na Podobieństwie (*Similarity-Based Crossover*,

SBX). Operatory te mają na celu dostosowanie odległości między rozwiązaniami rodzicielskimi a potomnymi zgodnie z wybraną miarą niepodobieństwa. Skuteczność tych operatorów jest oceniana w odniesieniu do standardowych operatorów w serii eksperymentów obliczeniowych. Operator TSM osiąga w większości przypadków lepsze wyniki niż standardowy operator mutacji, podczas gdy proponowane operatory krzyżowania osiągają wyniki porównywalne do standardowych.

W kolejnym wątku badawczym miary niepodobieństwa zastosowano w algorytmach sterowania różnorodnością, które są metodami mającymi na celu zwiększenie szerszego eksplorowania przestrzeni rozwiązań. Badanie porównuje standardowy algorytm ewolucyjny z niszowaniem (w wersji lokalnej i globalnej), poszukiwaniem nowości (*novelty search*, w wersji lokalnej i globalnej) i Algorytmem Genetycznym Sortowania Niezdominowanego II (*Non-dominated Sorting Genetic Algorithm II*, NSGA-II) wykorzystującymi różne miary niepodobieństwa. Analiza wyników pokazuje, w jaki sposób różne algorytmy i miary niepodobieństwa wpływają na wynik ewolucji pod względem jakości i heterogeniczności rozwiązań.

List of abbreviations used in the work

3D	Three-dimensional
DMX	Distance minimizing crossover
DPX	Distance preserving crossover
EA	Evolutionary algorithm
ED	Evolutionary design
EMD	Earth mover's distance
EMX	Equidistance minimizing crossover
FFT	Fast Fourier transform
GA	Genetic algorithm
GP	Genetic programming
SBX	Similarity-based crossover
TSM	Targeted sequential mutation

Contents

1 Evolutionary Design

"Evolution, to me, is the best designer of all time."
– **Frances Arnold**, Nobel laureate in chemistry

1.1 Introduction

Evolutionary design is a method of using evolutionary optimization techniques to design structures, in particular three-dimensional ones, for various purposes. This approach has several advantages over human designers. It doesn't require specialized knowledge, it can reduce the cost and time of the design process, and it avoids the *Einstellung effect* [19], which is a cognitive bias that causes experts to overlook better solutions due to their prior knowledge and assumptions. Evolutionary optimization methods can explore the solution space more extensively and discover optimal solutions that are beyond human imagination.

However, the full potential of these algorithms has not yet been realized. Evolutionary design is one of the most challenging optimization problems as the search space is usually infinite and combines both continuous and discrete aspects. The evaluation of solutions is costly, non-deterministic, and multi-objective. Furthermore, defining a genetic representation and operators for designs is not straightforward, and there are no clear guidelines on how to create representations with desirable properties. The complex and non-obvious representations that are often used in evolutionary design problems, along with their complex genotype-to-phenotype mapping, contribute to highly rugged fitness landscapes [16, 89], which hinder the search process.

This work aims to address these challenges by utilizing dissimilarity measures for 3D structures to enhance genetic operators and to explore the solution space more effectively. The thesis is organized as follows. In Chapter 1, the basic principles of evolutionary design are explained, and relevant literature is reviewed. Chapter 2 briefly describes the Framsticks simulation environment [124, 126] used in this study, and presents the dataset of active and passive 3D structures created for further analyses and experiments. Chapter 3 introduces the developed dissimilarity measures for 3D structures and provides their comparison. In Chapter 4, the dissimilarity measures are employed in the fitness-distance correlation (FDC)[103] analysis of selected evolutionary design problems. Chapter 5 presents genetic operators that utilize the developed dissimilarity measures for 3D structures. Finally, Chapter 6 analyzes the impact of different diversity maintenance algorithms and

dissimilarity measures on the fitness and diversity of the solutions. Chapter 7 concludes the work by summarizing the main contributions and providing directions for future research.

The rest of this chapter is structured as follows: Sect. 1.2 describes the definition and principles of evolutionary algorithms. Sect. 1.3 introduces the properties of genetic representations for these algorithms. Sect. 1.4 provides an overview of the work in the area of evolutionary design of novel structures.

1.2 Evolutionary algorithms

Evolutionary algorithms are optimization methods that draw inspiration from the process of biological evolution [8, 45, 191]. They operate by exploring the solution space of a problem using a population of solutions and mechanisms such as reproduction, mutation, recombination, and selection. While they are not guaranteed to find the global optimum, evolutionary algorithms can often provide high-quality solutions within a reasonable time. Therefore, they can be applied to optimization problems that can not be solved by other methods. Moreover, among the metaheuristics that were used for automated design, evolutionary algorithms turned out to be the most successful ones [16].

Genotypes and phenotypes

In evolutionary algorithms, solutions (phenotypes) cannot be directly manipulated, therefore a genetic representation is needed to encode them. The encoded version of the solution (phenotype) is called a genotype. During the course of the evolution, each phenotype is evaluated and assigned a numerical value that expresses its quality – the fitness value. This value is assigned to a phenotype, however, genetic operators such as mutation and crossover are applied to a genotype. Thus, the way in which a genotype is translated into a phenotype, the genotype-to-phenotype mapping, can affect the search process. For some simple problems, e.g. a knapsack problem [171], a genotype corresponds directly to a phenotype. However, evolutionary design problems usually employ genetic representations with a complex genotype-to-phenotype mapping that can increase the problem's difficulty. Fig. 1.1 illustrates the relationship between genotype, phenotype, and fitness spaces. The genetic representation and its operators introduce a certain topology of the phenotype space that in turn can strongly affect the fitness landscape.

Mutation, crossover, and selection operators

There exist various genetic operators that can be applied in evolutionary algorithms to manipulate the solutions in the population. Among them, mutation and crossover are the most common ones [200]. The aim of the mutation operator is to introduce diversity in the population and avoid premature convergence to a suboptimal solution. This operator usually randomly modifies, adds, or deletes a small part of a genotype to create a mutant solution. The crossover operator works by recombining the genetic information of two or more parent solutions to generate new offspring solutions that will inherit and combine some features of their parent solutions.

To choose the solutions that will survive and reproduce, resulting in the next gener-

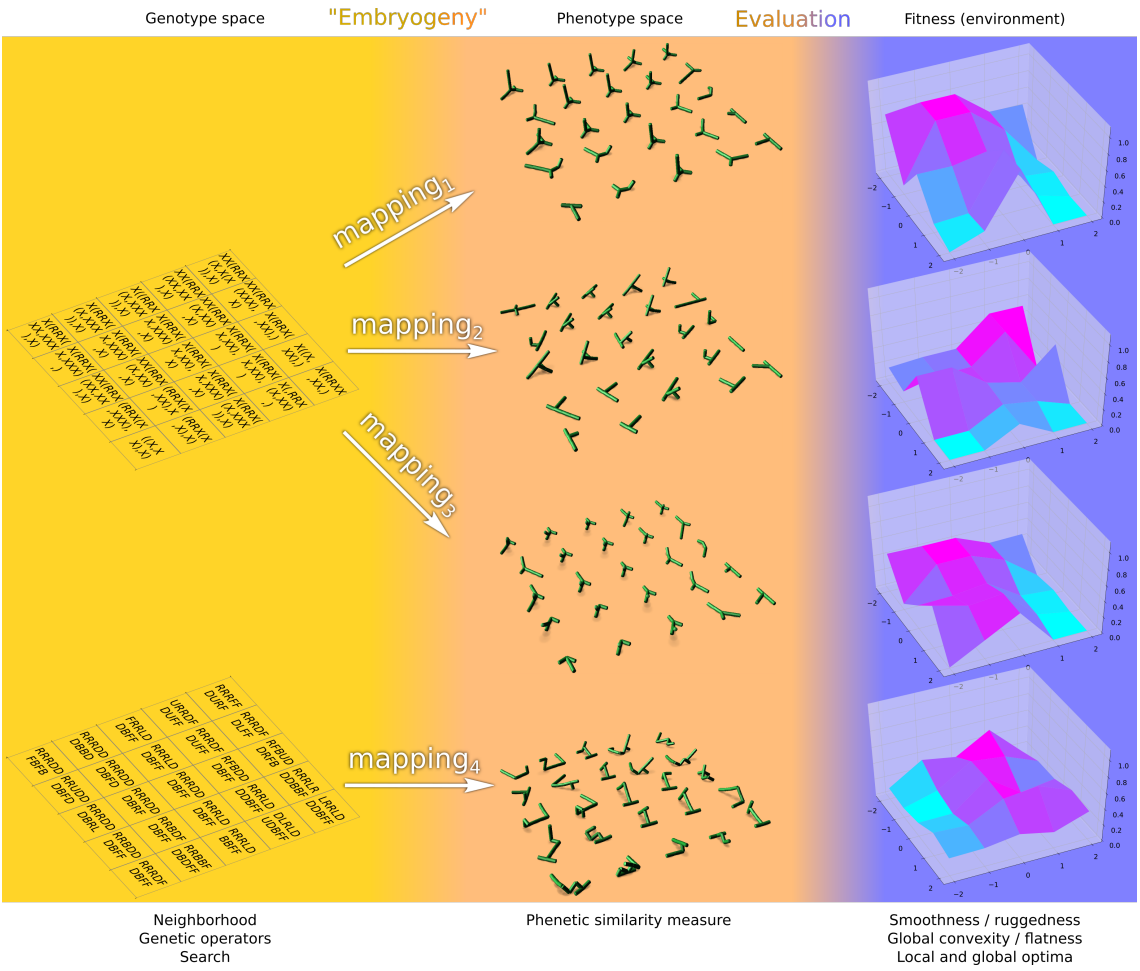


Figure 1.1: The relationship between genotype, phenotype, and fitness spaces in the problems of evolutionary design [117]. The upper row shows a sample genetic encoding of the solutions and its three genotype-to-phenotype mappings, *mapping*$_1$, *mapping*$_2$, and *mapping*$_3$. The lower row shows another genetic encoding and its genotype-to-phenotype mapping, *mapping*$_4$. Each of the mappings results in different topology of the phenotype space, which leads to a different fitness landscape. Note that the search process and genetic operators are applied in the genotype space.

ation, a selection operator is needed. It usually operates based on the solutions' fitness function value. This operator applies the selective pressure, which drives the algorithm towards better solutions [7]. There are various selection schemes that can be used, such as fitness proportionate selection, tournament selection, truncation selection, rank-based selection, and elitist selection [57]. The selection scheme affects the diversity of the population and the convergence rate of the algorithm.

The general flow of an evolutionary algorithm

There are many types of evolutionary algorithms, including:

- genetic algorithms [81, 82, 69],
- evolutionary programming [60, 59],
- evolution strategies [181, 8],
- and genetic programming [128].

The architecture of an EA can vary depending on the type and the specific implementation. The general scheme of a basic EA is shown in Fig. 1.2. It consists of the following steps:

1. Start by initializing a population of solutions randomly.
2. Evaluate each solution's fitness according to the fitness function.
3. Check if a stop condition is met, such as achieving a target fitness value or reaching a maximum number of iterations. If so, terminate the algorithm and return the best

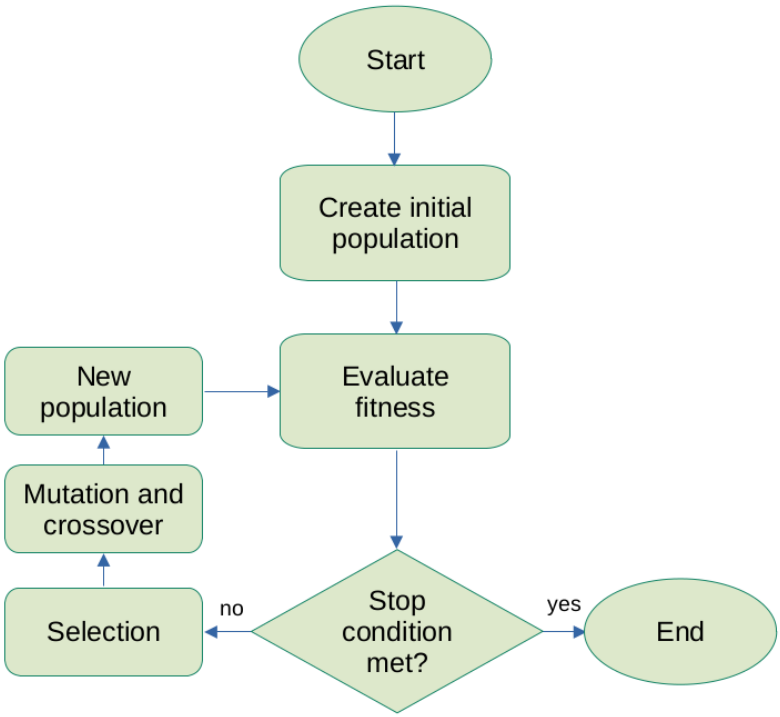


Figure 1.2: The general schema of an evolutionary algorithm.

solution. If not, proceed to the next step.

4. Select solutions from the current population based on their fitness values or other criteria.
5. Apply genetic operators, such as crossover and mutation, to the selected solutions to generate new solutions.
6. Create a new population.
7. Go back to step 2.

This process is repeated until the stopping condition is satisfied.

1.3 Representations for evolutionary algorithms

The genetic representation, its associated genetic operators, and the genotype-to-phenotype mapping are major factors that can influence the fitness landscape and the difficulty of the problem, as discussed in the previous section. This section presents the formal definition of genotype-to-phenotype and phenotype-to-fitness mapping proposed by F. Rothlauf [187] and introduces some concepts and properties that are useful for analyzing representations for evolutionary algorithms. The next section reviews the evolutionary design applications and the genetic representations and operators they use.

1.3.1 Genotype-to-phenotype and phenotype-to-fitness mapping

Rothlauf introduced a general framework for analyzing problems solved by evolutionary and genetic algorithms, based on the decomposition of the fitness function into two mappings: a genotype-to-phenotype mapping f_g and a phenotype-to-fitness mapping f_p [187]. He uses the following notation:

- Φ_g – the genotype space, which contains all possible genotypes x^g that can be encoded by the genetic representation,
- Φ_p – the phenotype space, which contains all possible phenotypes x^p that can be decoded from the genotypes by applying f_g,
- $f_g(x^g) : \Phi_g \to \Phi_p$ – the genotype-to-phenotype mapping, which transforms a genotype $x^g \in \Phi_g$ into a phenotype $x^p \in \Phi_p$,
- $f_p(x^p) : \Phi_p \to \mathbb{R}$ – the phenotype-to-fitness mapping, which evaluates a phenotype $x^p \in \Phi_p$ and assigns it a fitness value from $\mathbb{R}$.

The optimization problem can be generally defined as:

$$\hat{\mathbf{x}} = \max_{\mathbf{x} \in \Phi_g} f(\mathbf{x}),$$

where:

- $\mathbf{x}$ is a vector of decision variables (or another form of a genotype),

- $f(\mathbf{x})$ is a fitness function that combines the two mappings as follows:

$$f = f_p \circ f_g = f_p(f_g(x^g))$$

The function f_p assigns a fitness value to a phenotype $x^p \in \Phi_p$, while the genetic operators, such as mutation and crossover operate on its genotype, $x^g \in \Phi_g$. The translation between the genotype and the phenotype is determined by the genotype-to-phenotype mapping of the representation.

1.3.2 Properties of genetic representations

Rothlauf's theory of genetic representations focuses on three properties: **redundancy**, **scaling** of alleles, and **locality** [187]. A representation is **redundant** when there are more genotypes than phenotypes, meaning that the same phenotype can be encoded by more than one genotype. The redundancy of the representation can be either synonymous or non-synonymous. Synonymous redundancy means that the genotypes that map to the same phenotype are similar and can be easily changed into each other by genetic operators such as mutation and crossover. Synonymous redundancy can be either uniform or non-uniform. Uniform redundancy means that each phenotype has the same number of genotypes. Non-uniform redundancy means that some phenotypes have more or less genotypes than others. Synonymous and uniform redundancy does not affect the behavior of evolutionary algorithms. The performance of the algorithm in the case of synonymous non-uniform representation depends on whether the optimal solution is over- or under-represented. Non-synonymous redundancy hinders the evolutionary search by creating a misleading fitness landscape [187].

Scaling refers to the way in which alleles within a genotype are mapped into a phenotype. In uniform scaling, all alleles (building blocks) affect fitness equally. Conversely, in non-uniform scaling, some alleles have more or less influence than others. Non-uniform scaling can slow down the convergence and make the algorithm prone to genetic drift. **Locality** refers to the correspondence between genotype and phenotype spaces. In high-locality representations, similar genotypes produce similar phenotypes. High-locality representations preserve problem difficulty. In low-locality representations, a small change in a genotype can cause a large change in a phenotype. Therefore, low-locality representations can hinder the performance of the algorithm, making the search process more random [187].

Other properties useful for analyzing genetic representations are epistasis [34], evolvability [182], and scalability [86]. **Epistasis** is a concept denoting the interdependence of genes within a genotype. It can be quantified by different methods, depending on the type of genetic representation. High epistasis implies that there are strong interactions between genes and that a minor change in a genotype can cause a major change in a phenotype, resulting in a rugged fitness landscape. **Evolvability**, in the context of genetic representations, is in general understood as the ability to produce offspring that is advantageous for the evolutionary search. **Scalability** refers to the capacity of a representation to scale up to create more sophisticated designs.

Another framework for investigating genetic representations for ED problems in terms

of modularity, regularity, and hierarchy was introduced by Hornby [85]. **Modularity** of genetic representation enables a group of genes or alleles to be handled as a unit. **Regularity** allows for the reuse of the elements of the genotype. **Hierarchy** refers to the number of nested modules. These properties enhance the evolvability and scalability of representations by enabling reuse, recombination, and information compression.

1.4 Review of evolutionary design problems and representations

1.4.1 Scope of the review

Evolutionary design covers a wide range of applications that are useful for various stages of design, from optimizing existing designs to generating novel structures from scratch. These different tasks usually require different types of genetic encodings. In general, the genetic representations for evolutionary design problems can be classified into parameterizations and open-ended representations [84]. Parameterizations consist of fixed-length sets of values that allow for optimizing certain aspects of existing structures, such as element length or shape. They are suitable for tasks that involve modifying the predefined structure topology.

Open-ended genetic encodings, on the other hand, allow the generation of novel structures from scratch, without relying on a predefined structure topology. This work focuses on the latter application, therefore this review, putting an emphasis on the design representations and their genetic operators, only includes the works that employ open-ended genetic encodings. However, to showcase the various applications of ED, Table A.1 in Appendix A lists some works that were excluded from the review.

The analysis of applications of the included works allowed to determine the main domains: artificial life, generic design, robotics, engineering, and architecture. In the following section, the works are grouped by domains, and within each domain they are presented in chronological order.

1.4.2 Approaches to evolutionary design of novel structures

Artificial life

In his pioneering work, Karl Sims [196] introduced a system for evolving virtual creatures in simulated 3D environments. The genetic representation was based on directed graphs, used to describe both the morphology and the neural network of the creatures. The nodes of the graphs represented components such as body parts (segments), effectors, sensors, and neurons. The edges of the graphs represented either joints between body parts or synaptic connections between neural components. The mutation was performed by randomly adding, deleting, or modifying nodes or edges. The sexual reproduction was performed using one-point or multi-point crossover or using grafting. The creatures were evolved for the following tasks: swimming, walking, jumping, and following. The emergence of effective locomotion strategies, in some cases non-obvious from the point

of view of a human designer, showed the ability of the evolutionary algorithms to design virtual creatures.

Lohn and Reggia [152] investigated the potential of applying a genetic algorithm to generate self-replicating cellular automata. The genetic representation for automata consisted of an automata rule table. The single-point crossover was performed by swapping the partial rule tables from two parents. The mutation operator created a random action for a randomly chosen rule. The fitness function was based on a growth measure and a self-replication measure. The resulting self-replicating structures were better in terms of simplicity compared to those created manually. Furthermore, the structures displayed some surprising features, such as moving while replicating.

In [56] Eggenberger et al. introduced the concept of differential gene expression, inspired by the process of interaction between biological cells, regulating the organism's growth. This indirect developmental encoding was motivated by its ability to reduce the amount of information that has to be encoded in the genome compared to direct encodings, its ability to produce emergent properties, and the potential for parallel computation. The model included developmental processes such as cell division, death, and differentiation. The genotype consisted of a list of integers, divided into structural and regulatory genes. The mutation operator altered digits in the genotype. The crossover operator performed a single-point crossover. The optimization goal of the evolutionary experiment was to evolve bilateral structures with the fitness function measuring the number of cells, their x-axis position, and their symmetry. The paper demonstrated how biologically-inspired developmental encoding can be applied to the evolutionary design of 3D shapes.

Bonabeau et al. [22] proposed a model of growing 3D architectures using a multi-agent system inspired by social insects. The genetic representation used a set of micro-rules that guided the behavior of simple agents that could move and deposit blocks. The fitness function consisted of several criteria, such as the proportion of micro-rules used, the compactness of the architecture, and the patterns detected in the construction graph. Mutation exchanged an unused micro-rule with a randomly generated new one. In addition to a standard two-point crossover, a specialized crossover operator was devised. The micro-rules were divided into two sets minimizing the co-dependency between the sets. The crossover operator kept together mico-rules from the same sets. Among the resulting structures the amount of interesting shapes was higher than in random populations, but high epistasis between genes, resulting in a rugged fitness landscape hampered the ability of the model to discover more complex structures.

Komosinski and Rotaru-Varga [123] investigated how the use of different genetic encodings influences the evolution of virtual 3D creatures. They employed the Framsticks environment [124, 126] that allows for simulation of the active and passive 3D structures, modeled as undirected graphs. In the study, three genetic representations were compared: a low-level direct one, a recurrent one, and a developmental one. Each representation had its dedicated mutation and crossover operators. In the direct low-level encoding, the mutation operator altered exactly one element of an agent, by addition, deletion, or modification of its parameters. The crossover operator for this representation was performed by cutting and grafting the phenotypes. In recursive encoding, the mutation operator added or deleted a body part, a neuron, or a modifier or changed the parameters of a neuron. The crossover operator swapped the substrings of the parent genotypes. In developmental

encoding, the mutation operator affected a single node of the genotype tree, similar to recurrent encoding. The crossover operator swapped subtrees of the parent genotype trees. The agents were evaluated on three optimization tasks: the height of passive structures, the height of active structures, and the speed of locomotion. The low-level encoding produced solutions of lower fitness than the two higher-level encodings. The advantages of the recurrent and developmental encodings over the low-level one included a bias towards structured phenotypes and less disruptive genetic operators.

Generic design

Another early attempt at the evolutionary design of novel structures belongs to Bentley and Wakefield [17]. They proposed a genetic representation that used spatial partitions in the shape of cuboids with varying dimensions and locations. This representation was employed in a standard GA with real coding to evolve tables. The objectives included size, mass, stability, flat surface, and supportiveness and stability. Later [18] the same authors used a spatial-partitioning representation with optional planes intersecting the cuboids. Genotypes were encoded as binary strings that specified the number, size, position, orientation and type of each primitive. The mutation operator added or deleted blocks of genes corresponding to the primitives. A hierarchical crossover was employed. This approach allowed to evolve a variety of solid objects including tables, heat sinks, optical prisms, and streamlined designs like train fronts.

Husbands et al. [92] applied genetic algorithms to an interactive design of 3D objects. Their approach was based on a shape description language, utilizing superquadric primitives and their deformations. The genotype representation was a directed network, traversed recursively in order to produce an expression of a shape descriptor language. The genetic operators operated on the bitstrings encoding the networks. The mutation, performed bitwise, could modify the structure of the expression or alter its parameters. As a crossover operator, the two-point crossover was used. The selection was performed by users. As a result, varied and sometimes unexpected shapes were generated.

Broughton et al. [27] employed 3D shape grammar and genetic programming to interactively evolve 3D structures. The genotypes were represented using program trees. The terminal set consisted of a box, cone, cylinder, sphere, torus, and wedge. The function set included transformations such as growth, rotation, or displacement. Crossover was performed by swapping the subtrees between two parents. Mutation replaced a randomly selected subtree with a newly generated one. The selection was performed by users, according to aesthetic criteria. The system's goal was to explore the solution space rather than optimize shapes.

In [88] Hornby and Pollack compared the generative representation based on parametric, context-free L-systems to a non-generative encoding in the task of evolving tables. The objects were built from voxels. The mutation operator modified a command or its parameters, while the crossover operator inserted a small segment of one parent into a copy of the other parent. The fitness function comprised a table's height, surface structure, stability and the number of excess voxels used. The generative representation proved to provide better results than a non-generative one both in terms of the fitness of the solutions and execution time.

Ebner [55] proposed a method of evolving three-dimensional shapes using scene graphs, which are hierarchical data structures that describe the objects and their transformations in a scene. He compared two different scene graph representations: the one used by OpenInventor [224], a 3D graphics toolkit, and the one used by virtual reality modeling language (VRML) [1]. The scene graphs consist of nodes that can be either shape nodes, which specify the shape type, property nodes, which specify the shape look, transformation nodes, which specify translation, rotation, or scaling operations, or group nodes. Mutation defined in the study for a scene graph representation replaced a random node with a newly generated subtree. The crossover operator swapped the selected subtrees from two parents. The representation was evaluated on the optimization of the blades of a wind turbine. The fitness of the solutions was calculated as the average rotational energy of the rotor. The VRML representation turned out to produce better results. The OpenInventor representation using group nodes was disadvantageous as the genetic operators could lead to significant differences in the resulting phenotype.

Hamza and Saitou [77] proposed a method for evolving three-dimensional solid shapes using constructive solid geometry (CSG). They encoded the shapes as binary CSG trees, where the internal nodes were boolean operations (union or subtraction) and the leaf nodes were solid primitives (cube, cylinder, or sphere). The crossover operator swapped subtrees of two parents. Mutation changed a boolean operation or a primitive solid or altered the parameters of the solid. The representation was evaluated on the problem of the design of indoor modular space truss joints, which are structures that connect beams and columns in buildings. The fitness function was based on two objectives: maximizing stiffness and minimizing weight. The selected resulting structures were later fabricated.

O'Neill et al. [165] employed a grammatical evolution approach to evolve 2D shapes comprised of pixels. A shape grammar was used to generatively construct the phenotypes. The operators of the grammar included such actions as moving the pixels, changing their sizes, and drawing shapes (e.g. a circle). A genotype was represented using integer codon values. Random resetting mutation operator and single-point crossover were used. Fitness was calculated as a distance to the target structure. Three targets, differing in level of difficulty, were considered. In all three cases, a correct solution was obtained.

Kiptiah et al. [112] used interactive genetic algorithms to design 3D models, rendered in Blender. The models were encoded using L-systems. The terminal symbols set included such operations as creating a branch segment, creating a quad, or rotating around a given axis. Standard GA mutation and crossover operators were used. The fitness of the models was determined by the users' preferences, who interacted with the system through a graphical user interface. However, the system faced a major challenge in providing specific feedback to guide the evolution towards models with desired properties.

Robotics

In [62] Funes and Pollack evolved 2D and 3D Lego brick structures in the context of evolutionary robotics. The structures were represented using trees, where a node corresponded to a brick and had a parameter denoting the size and a list of descendants. Crossover swapped subtrees between two parents. Mutation either modified a brick parameter or added a node. In the experiment concerning 3D structures, the goal was to evolve a table

and the objectives included the desired height, coverage of the target area, support of the desired weight, and minimization of the number of bricks used. The objects evolved in the simulation were later physically built, which demonstrated the possibility to use evolutionary computation to design physical objects.

In [147] Lipson and Pollack co-evolved from scratch morphologies and control systems of the simple robots, consisting of bars, actuators, and artificial neurons, with the goal of locomotion. The evolved robots were later built in a physical world. The robots were represented using a string of integers and floating point numbers describing the body parts, neurons, and their connections. The mutation operator changed the properties of bars or neurons, removed or added bars or neurons, or changed their connectivity. The crossover operator was not used. Fitness was defined as the distance traveled by center-of-mass in a fixed time interval. The experiment resulted in robots varying in morphologies and locomotion strategies, and the emergence of unexpected properties like symmetry.

Hornby, Lipson, and Pollack [87] co-evolved morphologies and control systems of robots built of bars and fixed and actuated joints. The evolved robots were later built in a physical world. Fitness was again the distance traveled by the center-of-mass. Parametric, context-free L-systems were used as a genetic encoding. Mutations included such operations as replacing, adding, or removing commands and altering their parameters. The crossover operator worked by inserting a small segment of one parent into a copy of another parent. In the experiments, several emerging locomotion strategies were observed. The generative encoding showed the advantage of allowing to re-use of parts of a genotype, enabling modularity.

In [24] Bongard and Pfeifer introduced an artificial ontogeny inspired by the process of biological development. The agents, equipped with sensors and actuators, were evolved in a virtual environment for a block-pushing task. The morphology and control system of the agents was co-evolved. The genotypes, directing the growth of the agents, were encoded using gene regulatory networks. The mutation operator altered, on average, a single value in a genotype. The crossover operator was an unequal crossover. The fitness function comprised the distance and speed of pushing the block. The experiments showed a dissociation between genotype and phenotype complexity. The evolved agents exhibited hierarchical repeated structure, indicating high evolvability of the developmental encoding.

Krĉah [130] introduced a method for the evolutionary design of autonomous robots, inspired by NeuroEvolution of Augmenting Topologies (NEAT) [204]. The algorithm was applied to co-evolve the morphology and control systems of simulated robots. The genetic representation – a directed graph – and its operators were based on those introduced by Sims [196]. Mutation randomly changed a node in the graph, either by modifying its parameters or by adding or deleting a child node. In the hierarchical recombination process, phenotypes were mated based on historical markings, where morphology graphs of parents were examined for nodes and connections with matching markings. Fitness sharing (utilizing compatibility distance based on the historical markings) was used as a mechanism of speciation. Fitness functions included light following, swimming, walking and, jumping. For all four fitness functions, the proposed hNEAT algorithm outperformed the standard GA. The speciation mechanism allowed to avoid premature convergence to local optima. The recombination based on historical markings enhanced the preservation of advantageous features, protecting against disruptive changes.

Faíña et al. [58] applied a generative evolutionary approach to co-evolve the morphology and the control systems of simulated robots that were composed of heterogeneous parts. The robots were represented using tree-like structures, where each node corresponded to a module that had a specific type, control parameters, and connections to other modules. The mutation operator performed various operations on the nodes, such as adding, deleting, changing the parameter values, changing the connectivity, and replicating a branch to create a symmetrical structure. The crossover operator was not used in the evolutionary process, because the preliminary experiments showed that it had highly disruptive effects on the tree representation. The authors considered two different objectives in their experiments. In the first experiment, the objectives were to maximize the distance traveled by the robot while carrying a payload and to minimize the energy consumption of the robot. In the second experiment, the objective was to maximize the area painted by the robot. The robots constructed in the latter experiment were later fabricated.

Chee and Teo [35] used genetic programming and a self-adaptive differential evolution algorithm to co-evolve the morphology and control system of simulated snake-like modular robots. The robots were modeled using a tree-based representation, where each node corresponded to a segment of the robot and specified its properties, such as its length and connections to other segments. The GP crossover operator exchanged subtrees between two parent solutions. For the mutation, the self-adaptive Differential Evolution algorithm was used. Two mutation operators were compared: a standard differential evolution mutation and its customized version, differing in the mutation differential operation. The objective of the evolutionary experiment was to achieve the forward locomotion of the robots along a straight line. The results showed that the customized DE mutation operator provides better results than the standard operator.

Brodbeck et al. [26] introduced an approach to the artificial evolution of physical robots, employing a "mother robot" that designed and assembled the agents. The fabricated robots were evaluated in the testing environment and their performance was used in the development of the new generation. The genetic representation of the agents consisted of a variable-length sequence of genes, each encoding information about a module type, construction parameters, and motor control. The genetic operators included mutation – which modified, added, or removed a single gene – and a single-point crossover. The fitness value was calculated based on the distance traveled from the robot's initial position in a specified time interval. The experiments showed the emergence of the agents diversified both in terms of morphologies and locomotion strategies. The analysis of the physical construction constraints on the resulting morphologies revealed the strong influence of those constraints on the phenotypic diversity.

Cheney et al. [148] analyzed the reasons hampering the co-evolution of morphologies and control systems of virtual creatures. The experimental analysis employed 3D soft robots. The robots were modeled as a grid of voxels, represented using a directed graph, following the Compositional Pattern Producing Network Neuro-Evolution Through Augmenting Topologies (CPPN-NEAT) algorithm [203]. Two separate networks for morphology and control system were employed. Each non-empty voxel could be either passive or active. The mutation operator could add or remove a node or an edge, modify the weight of an edge, or modify a node's activation function. The crossover operator was not used. The fitness was calculated as a displacement of the agent's center of mass along the x-axis.

The results showed that the morphologies converge faster than the control systems, which limits the exploration of novel behaviors. The possible explanation for this phenomenon may be the more rugged fitness landscape of morphologies in comparison to that of control systems.

Talamini et al. [207] explored the problem of designing adaptable morphologies of soft robots. The robots were represented using a grid of voxels. Gaussian mutation operator and uniform crossover were used. The proposed task-agnostic approach employed criticality as a fitness function. A dynamical system possesses the property of criticality when it is close to a phase transition between the ordered and chaotic states. The measure of criticality that was used is based on the avalanche analysis. Three different tasks were used to evaluate the evolved robots. The robots optimized using criticality were more adaptable than handcrafted robots, and they performed well across different tasks, unlike the handcrafted robots that performed poorly in the tasks they were not designed for.

Engineering

Schoenauer [192] proposed a genetic representation for shape optimization based on Voronoï diagrams, which are a mathematical construct that partitions space into regions according to the distance to a set of points, known as Voronoï sites. Each region, called a Voronoï cell, contains all the points that are closer to its corresponding site than to any other site. The collection of cells forms a Voronoï diagram that covers the whole space. The genetic representation consisted of a list of Voronoï cells with labels 0 or 1. A random line that crossed a Voronoï diagram was used for performing crossover, and the sites on either side of the line were swapped between two parents. Mutation randomly changed the location or label of a site, or added or removed a site. This representation was compared to H-representation, in which the topology of the structure is defined by the list of rectangular "holes" with given locations and sizes. Crossover for H-representation worked similarly, by geometrical exchange of holes. Mutation modified the location or size of the holes or added or deleted them. For both representations, the fitness function was computed by comparing the evolved shape with the target shape. Both Voronoï diagrams and H-representation outperformed bitarray representation. However, in the case of the topological optimum design problem, H-representation had an advantage over Voronoï diagrams due to its lower epistasis.

Baron and Fisher [14] applied voxel representation in shape optimization: The genotype was a binary string that encoded the presence or absence of voxels in a two-dimensional design space. The test problems were designing a beam and an annulus. The fitness function measured weight and stress. Initially, the bitwise mutation and the two-point crossover operators were used, but they proved to be inefficient in terms of the time required for significant improvement. Then, the smoothing mutation operator was tested, which replaced the values of neighboring voxels with their most common value. This operator was more effective, as it eliminated isolated holes and ragged edges in the designs. The UNBLOX [29] crossover operator was also employed, which swapped sub-grids instead of 1D sub-strings that did not fully represent the adjacency of cells in the 2D grid. The UNBLOX crossover outperformed standard crossover operators in terms of both speed of convergence and best fitness obtained. Two new mutation operators that exploited the

properties of the task were also devised. Both operators outperformed bitwise mutation. The specialized operators for the beam design problem were ineffective for the annulus design problem and had to be modified to incorporate domain knowledge. The paper showed that the voxel representation with specialized representation can be successfully applied to shape optimization.

Peysakhov and Regli [169] applied messy genetic algorithms to design structures composed of Lego elements. The genetic representation was based on assembly graphs, where nodes represented Lego elements and edges represented connections between elements. The labels assigned to nodes defined the type and the parameters of an element. A genotype consisted of an array containing nodes and a hash table containing edges. The mutation operator replaced a Lego element with an element of the same type and different size. As a crossover operator, a single-point crossover was employed. The goal of the experiments was to evolve structures of a certain type (e.g. pillar-like structure) exhibiting desired properties (e.g. a minimum number of holes). The experiments showed that the performance of the genetic algorithm was worse for structures represented by highly connected graphs, which could be explained by the highly disruptive effect of the crossover operator for this type of structures.

Kicinger et al. [109] proposed a method for the evolutionary design of steel structures in tall buildings using cellular automata (CA) as a generative representation. The genotype consisted of separate design embryos for different building subsystems and one-dimensional CA rules for each subsystem. The design rules were applied to corresponding design embryos. After the development, the subsystems were assembled into a complete design. The bit flip mutation operator, adapted to handle non-homogenous genotypes, and the uniform crossover operator were used. The fitness of solutions was calculated as the weight of the steel structure. This initial study showed that CAs are a feasible representation for the optimization of complex structures that exhibits the property of scalability.

Hornby et al. [83] used an evolutionary algorithm to design an antenna for NASA's Space Technology 5 (ST5) mission. They employed a generative tree-structured representation, where the nodes contained antenna-construction operators. The fitness value of the solutions was calculated as a function of the voltage standing wave ratio (VSWR) and gain values. The evolved antenna was later fabricated and deployed in the mission, proving the feasibility of the proposed approach.

Architecture

Rosenman [186] used an evolutionary approach to architectural form generation. A hierarchical growth model was employed: first, a population of rooms was created, next, zones were generated from the rooms. The architectural forms were represented by a sequence of rules of design grammar. The mutation operator was not used. Instead, the experiments were rerun with different initial randomly generated populations. A crossover operator was a single-point crossover that swapped the subsets of rules between parents. The fitness function involved both a quantitative component (including such factors as minimizing adjacency requirements between rooms) and a qualitative component, assessed by users. The bottom-up hierarchical approach showed the advantage of allowing for shorter genotypes and reducing combinatorial problems.

Similarly to [27], using aesthetic aspect as a selection criterion, O'Reilly and Ramachandran [167] employed an evolutionary approach to create 3D structures in the context of architecture. The genotype consisted of an initial form drawn by an architect and a series of transformations. Only the transformations were subject to the genetic operators: mutation and single-point crossover. The selection was performed by users.

O'Neill et al. [164] further developed the approach introduced in [165] and applied it to shelter design. A simple representation consisting of a list of beams and their coordinates turned out to not be able to capture the compositional character of a building. An improved representation was introduced that operated on curves and a list of beams associated with the points on the curves. The same mutation and crossover operators as in [167] were used. Fitness was assigned by users according to perceived aesthetic values. The authors noted that recursion in grammar is a desirable feature as it enabled to evolve complex objects, however excessive complexity resulted in incoherent structures composed of too many components.

Byrne [28] employed grammatical evolution to design bridges. The bridges were modeled as undirected graphs. The nodes were described by Cartesian coordinates and a description attribute denoting the part of the bridge the node belongs to. The fitness function was a multi-objective function that aimed to minimize two conflicting objectives: the stress experienced by the bridge under a given load, and the amount of material required to construct the bridge. The experiment showed that the proposed approach was able to produce solutions satisfying the specified constraints. However, the analysis of the users' preferences revealed that the Pareto-optimal solutions were less aesthetically pleasing for users than solutions that were not on the front.

Other

Lohn and Colombano [150] used evolutionary search to design analog circuits. A language comprised of component-placing instructions was used to encode the genotypes. The crossover operator was a single-point crossover. The phenotypes created from evolved genotypes were later simulated to assess the value of the fitness function. The objectives, optimized in separate experiments, included filter response and amplifier gain. The evolved circuits did not outperform their hand-designed counterparts, but the experiments showed that it is possible to automatically design feasible circuits characterized by desirable properties.

To improve the scalability of evolutionary circuit design, Gordon [71] applied an indirect developmental encoding, inspired by the process of biological development. Genotypes were modeled as sets of rules governing the gene expression. The rules consisted of preconditions and actions. Preconditions were boolean expressions that checked the presence of proteins needed for a gene to activate and the absence of its inhibitors. The developmental rules were applied to a two-dimensional arrays of cells that were later mapped to circuits. A point mutation and a single-point crossover operator were used. The optimization task was to evolve a two-bit adder, where the fitness functions was calculated based on a distance to a target pattern. As a result, functional circuits were evolved. The developmental representation showed the advantage of encoding design abstractions, such as iterative patterns, enhancing scalability.

Lameijer et al. [136] developed an interactive evolutionary approach for the design of drug-like molecules, which allows the user to explore the chemical space of all possible molecules. The molecules were represented as graphs, where nodes corresponded to atoms and edges corresponded to bonds. The nodes representing atoms of different valence were encoded using different symbols. The crossover operator swapped subtrees of two parents. The mutation operator altered the molecular structure by adding, deleting, or modifying atoms or bonds while ensuring that the valence rules were satisfied. The selection was performed interactively by users. The authors concluded that the employed atom-based mutation is more advantageous than the alternative fragment-based mutation, as in the latter approach the mutations result in phenotypes vastly different from their parents.

Baldominos et al. [11] used evolutionary algorithms to automatically design the topologies of convolutional neural networks (CNN) for handwritten digit recognition. Two different genetic encodings were used to represent the networks: the Gray encoding and a formal grammar. The genetic operators included random resetting mutation and multipoint crossover. Since the solutions tended to converge to a single solution, a niching strategy was implemented to promote diversity. The authors evaluated the performance of the evolved CNNs on the MNIST dataset, a benchmark for handwriting recognition. The results showed that the evolutionary approach is capable of designing networks that are competitive with state-of-the-art solutions. The formal grammar representation allowed for obtaining better results than Gray encoding and standard GA, which can be attributed to the greater flexibility and lesser redundancy of this encoding.

Year	Domain	Problem	Representation	Ref.
1994	Artificial life	Evolving virtual creatures	Directed graph	[196]
1995	Generic design	Table evolution	Spatial-partitioning representation	[17]
1995	Artificial life	Evolving self-replicating structures	Cellular automata	[152]
1996	Generic design	3D shape design	Directed network	[92]
1996	Engineering	Cantilever plate design	Voronoi diagrams	[192]
1997	Generic design	3D shape design	Spatial-partitioning representation	[18]
1997	Architecture	2D floor plan design	Design grammar	[186]
1997	Artificial life	3D shape design	Cell grid	[56]
1997	Generic design	3D shape design	Tree	[27]
1998	Architecture	Evolving architectonical forms	Profile set and a sequence of transformations	[167]
1998	Robotics	Evolving robots	Tree	[62]
1999	Circuit design	Evolving circuit designs	Set of rules	[150]
1999	Engineering	Beam design	Voxels	[14]
2000	Artificial life	3D shape design	List of instructions	[22]
2000	Robotics	Evolving robots	String of numbers	[147]
2001	Robotics	Evolving robots	L-system	[87]
2001	Generic design	Evolving tables	L-system	[88]
2001	Robotics	Evolving robots	Gene regulatory network	[24]
2001	Artificial life	Evolving virtual creatures	String	[123]
2003	Generic design	Evolving wind turbine	Scene graph	[55]
2003	Circuit design	Evolving circuit designs	Set of developmental rules	[71]
2003	Engineering	Evolving engineering designs	Assembly graph	[169]
2004	Generic design	Truss optimization	Tree	[77]
2004	Engineering	Design of steel structures in buildings	Cellular automata	[109]
2006	Engineering	Evolving antennas	L-system	[83]
2006	Chemistry	Design of drug-like molecules	Graph	[135]
2008	Robotics	Evolving robots	Directed graph	[130]
2009	Generic design	2D shape design	Shape grammar	[165]
2010	Architecture	Shelter design	Shape grammar	[164]
2012	Architecture	Bridge design	Shape grammar	[28]
2013	Robotics	Evolving robots	Tree	[58]
2014	Robotics	Evolving robots	Tree	[35]

Year	Domain	Problem	Representation	Ref.
2015	Robotics	Evolving robots	List of instructions	[26]
2016	Robotics	Evolving robots	Directed graph	[148]
2017	Generic design	3D shape design	L-system	[112]
2018	Neuroevolution	Evolving network topology	Grammar, Gray encoding	[11]
2021	Robotics	Evolving robots	Voxels	[207]

Table 1.1: Works in the evolutionary design reviewed in this chapter.

The papers reviewed in this section are summarized in Table 1.1. Fig. 1.3 presents the distribution of domains of the papers in the following years. It should be noted that this review is not representative of the entire body of work in the area of evolutionary design, as works employing parametric representations were excluded and the focus was put on creating novel designs from scratch. In this respect, there is an observed prevalence of work on artificial life and generic design in earlier years and on robotics in later years. This trend may reflect a difficulty in advancing effective search methods for generic evolutionary design problems.

Still, the reviewed works demonstrated the emergence of surprising behaviors or morphologies that would be difficult for a human designer to invent [196, 152, 92]. Some of the evolved structures were later physically built [62, 147, 87, 77, 58], proving the applicability of evolutionary design to industrial applications. For instance, the antenna evolved by Hornby [83] was successfully employed in the NASA mission.

Fig. 1.4 shows the distribution of the genetic encodings used in the reviewed works. The most frequent representations were L-systems and tree-based representations. In general, indirect, recurrent, and generative encodings, allowing for modularity and re-use of parts of the genotype, were more prevalent. Such encodings bias the search towards more structured phenotypes, allowing to discover morphologies of greater complexity. On the other hand, higher epistasis of this type of representations may lead to a lower locality of genetic operators and in turn a more rugged fitness landscape. The disruptive effect

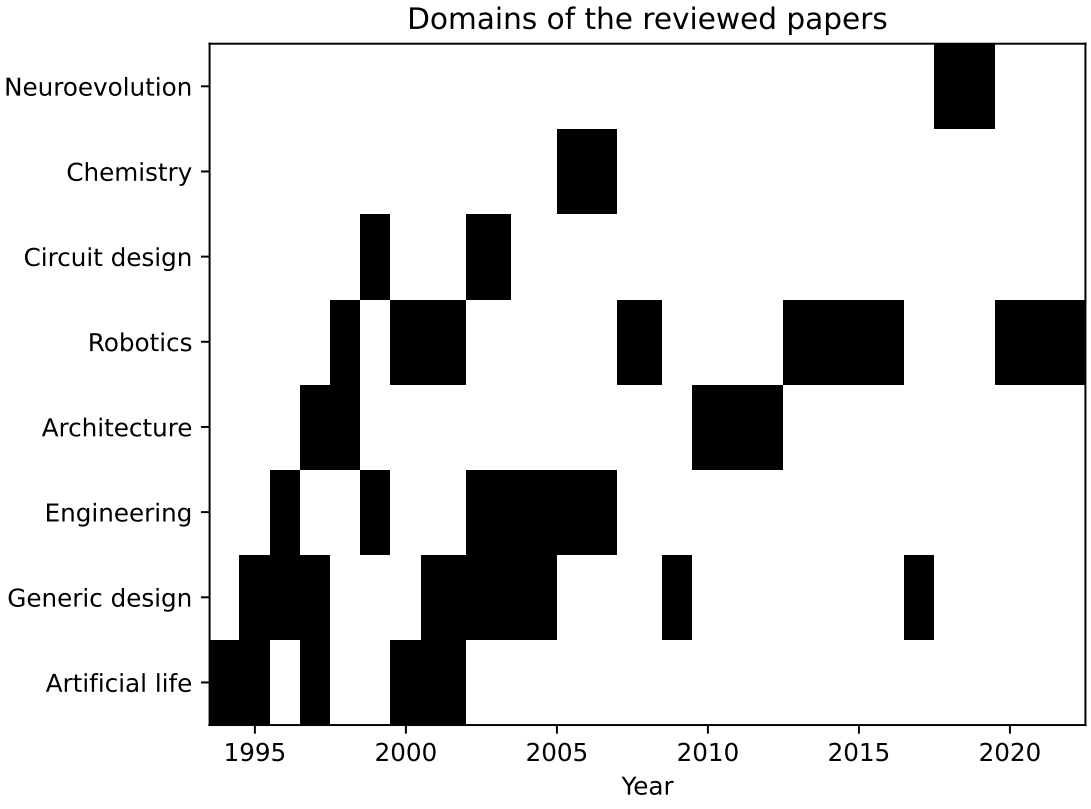


Figure 1.3: Domains of the papers featured in the review.

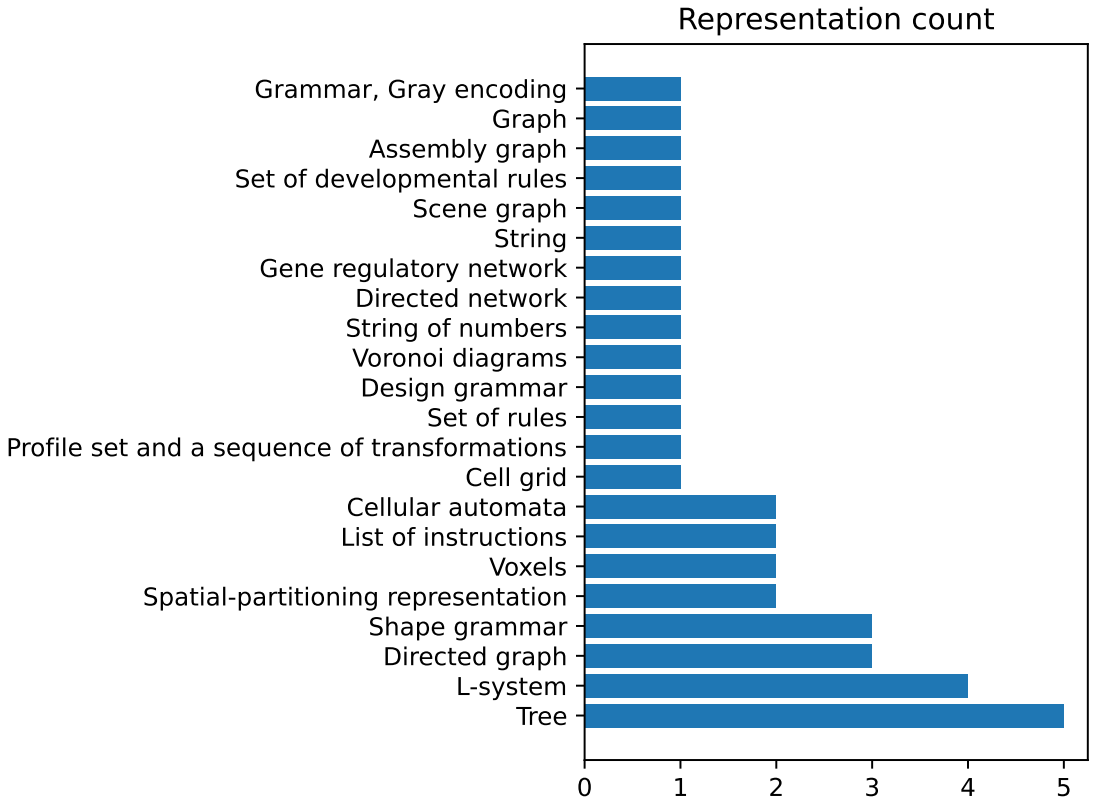


Figure 1.4: Representations used in the papers featured in the review.

of some genetic operators was reported in the reviewed works [192, 55, 169, 136] and in some works the crossover operator was not employed at all [147, 148, 58]. Conversely, the specialized operators taking into account properties of the representation turned out to increase the search efficiency [14, 130, 35].

Overall, the works reviewed in this section show that indirect encodings can provide the features of modularity, regularity, and scalability, which are desired in ED problems. However, these representations also pose challenges for the design of effective genetic operators, which should avoid disrupting the structural and functional properties of the parent solutions. Therefore, a promising approach is to use phenetic dissimilarity measures to control the amount of variation introduced by the genetic operators. The development of phenetic dissimilarity measures and their integration into the evolutionary search process is the main topic of the subsequent chapters.

2

Data set of active and passive 3D structures

2.1 Introduction

The experiments in this work were conducted using the Framsticks simulation environment [124, 126] and its Python interface [127], which enables the implementation of customized genetic algorithms. This chapter provides an overview of the Framsticks simulation environment and the 3D structure model that it uses to represent the artificial creatures (Sect. 2.2). Then, it describes the genetic representations available in Framsticks, with an emphasis on the ones that were employed in this work (Sect. 2.3).

For the analyses performed in the following chapters, a data set of active and passive 3D structures was constructed. The data set consists of structures encoded using three different genetic representations and evolved for four separate optimization objectives. The method of generation and selection of the structures for this data set and its properties are described in Sect. 2.4.

2.2 3D structure model

Framsticks software simulates and evolves three-dimensional designs controlled by recurrent neural networks. Two physics engines are available: a native MechaStick and ODE [197], which simulates rigid bodies composed of cuboids, ellipsoids, and cylinders. In the experiments reported here, a simple and fast MechaStick physics engine was used. The engine uses the finite element method: bodies are composed of points with mass that are connected with joints, and only the points are simulated. The points are referred to as parts. At each part, the simulator computes every force that acts on it: the pull of gravity, the springiness when connected to other points, the bounce and drag when hitting the ground, etc. (Fig. 2.1). Control systems consist of artificial neurons of various types, including sensors and effectors (Fig. 2.2), that can be connected arbitrarily – with recurrent and parallel connections permitted (for a detailed description of the neuron types, see [126]).

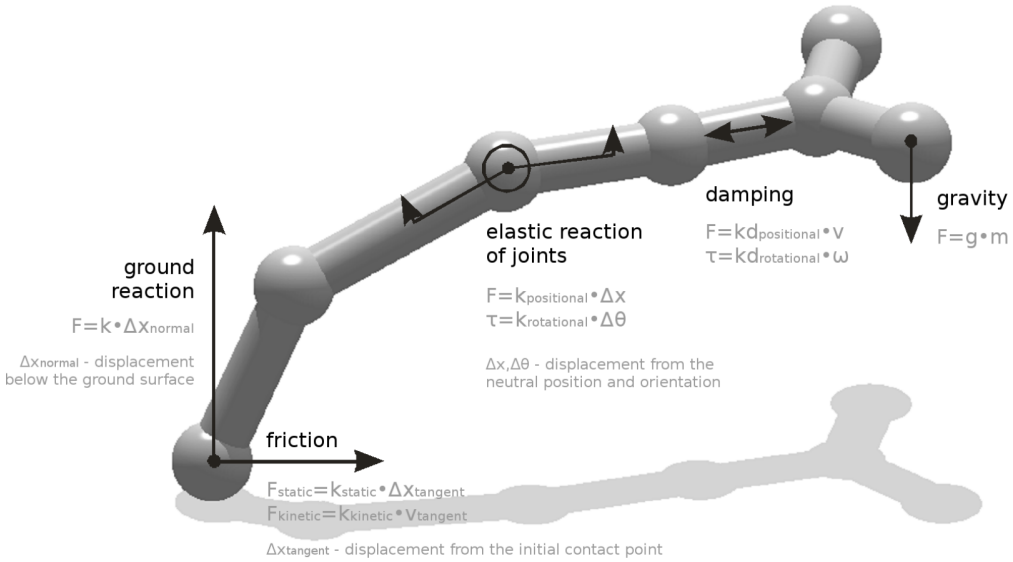


Figure 2.1: Sample forces calculated in the MechaStick simulator [216].

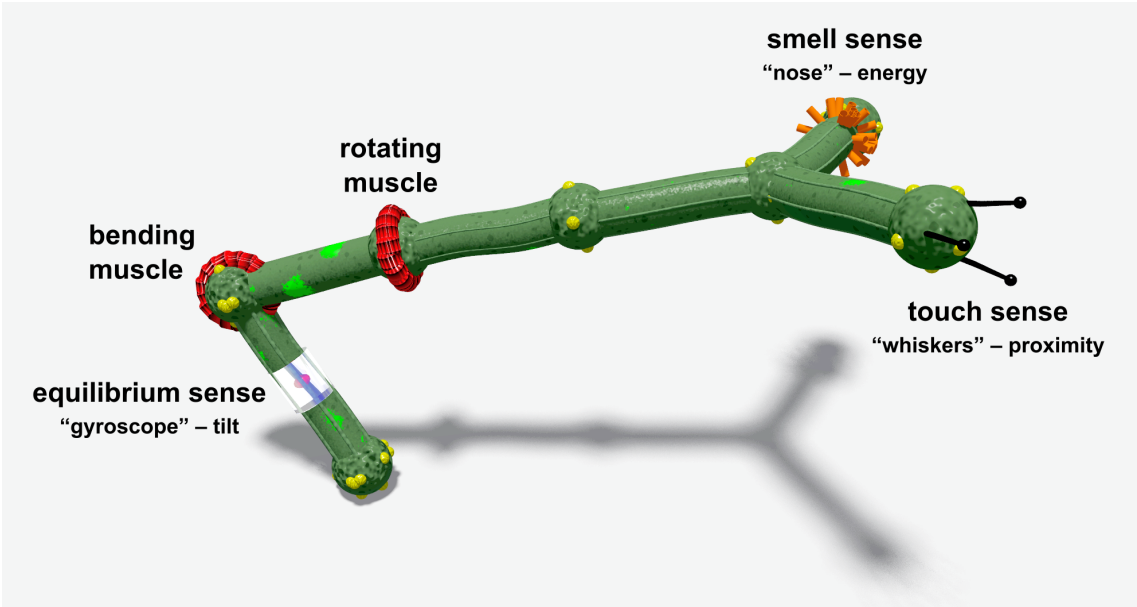


Figure 2.2: A visualization of a sample body simulated by the MechaStick engine with parts (points) as balls, joints as sticks between parts, and selected sensors and effectors. This particular structure has 7 parts and 6 joints [126].

2.3 Genetic representations in Framsticks

A genotype describes both a body and a control system, and it can undergo mutation, crossover, and repair. Framsticks offers various genetic representations with its dedicated operators for encoding active and passive 3D structures. Fig. 2.3 illustrates the hierarchy of these representations in Framsticks. The higher-level representations, such as *f1*, *f4*, and *fH*, need to be converted to one of the lower-level representations, either *f0* (using the ball-and-stick model) or *f0s* (using the solid shapes model), to generate a model that can be simulated. Fig. 2.4 shows examples of such models both inside and outside the simulation. The genotypes are encoded using strings of characters, as demonstrated in Fig. 2.5. The following list provides a brief overview of the genetic representations in Framsticks and their features:

- *f0/f0s*: a low-level encoding that enables the representation of any creatures,
- *f1*: a recursive encoding,
- *f4*: a developmental encoding sharing some symbols and concepts with *f1*,
- *fH*: a similarity encoding,
- *fB*: a biological encoding built on top of *fH*,
- *f7*: a messy encoding that accepts any string of symbols,
- *fL*: a parametric Lindenmayer system (L-system [174]),
- *f9*: a 3D turtle encoding,
- *fF*: a foraminifera parametric encoding,
- *fS*: a solid shape encoding inspired by *f1*,
- *fG*: a gene regulatory network encoding,
- *fn*: an encoding for numerical optimization that consists of a vector of real numbers.

This work uses three different types of encodings: a direct one (*f0*), a direct recurrent

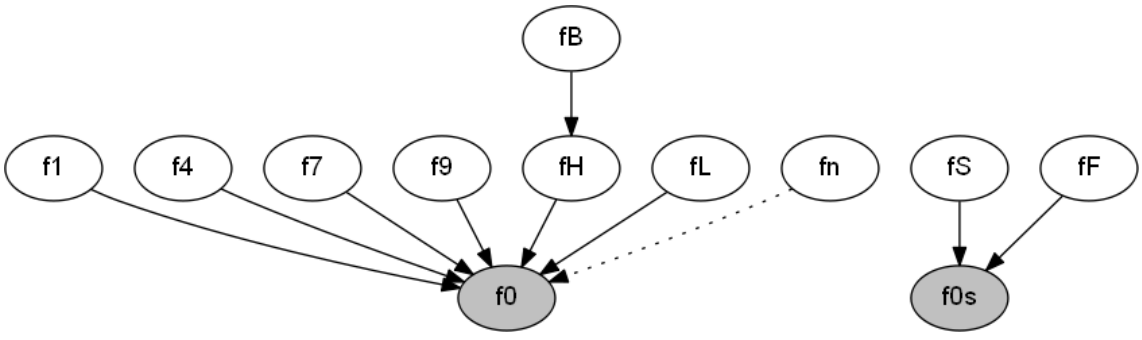


Figure 2.3: The genetic encodings in Framsticks and their hierarchy. The encodings are shown in ellipses. Higher level encodings require conversion to lower level encodings, as indicated by arrows. The gray ellipses denote two direct encodings: *f0* (using ball-and-stick model) and *f0s* (using solid shapes model). The dashed line corresponds to *fn*, which is an encoding for representing real numbers that are used in separate numerical optimization experiments [126].

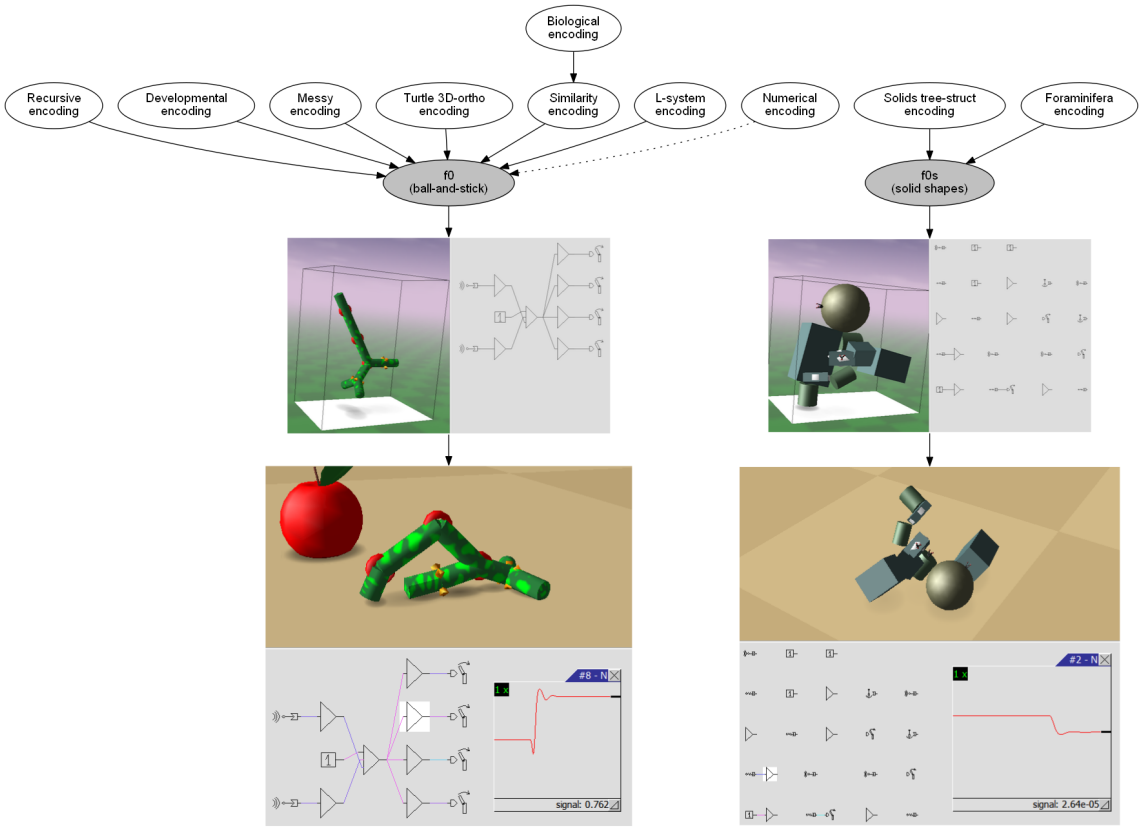


Figure 2.4: The descriptive names of the encodings (upper row), the two model types: ball-and-stick and solid shapes (middle row), and simulated structures (bottom row) [126].

one (*f1*), and a developmental one (*f4*). The following sections describe them in more detail.

2.3.1 Direct encoding (*f0*)

Syntax

The *f0* is a low-level direct encoding in which each line describes one object from the model and its parameters. The general syntax is as follows [126]:

```
CLASSID:PROPERTY1=VAL1, PROPERTY2=VAL2, ...
```

where `CLASSID` is an alphanumeric identifier of one of the four classes of objects. Each class has its own properties, specified after the colon. The `PROPERTY=` can be omitted if the given property is next in sequence. The attributes of an object do not have to be specified if their values are equal to the default. The properties of the classes are as follows:

- **Part** (`p:`)
  - **position** (`x, y, z`) and **orientation** (`rx, ry, rz`) in 3D space,
  - physical properties: **size** (`s`), **density** (`dn`), and **friction** (`f`).

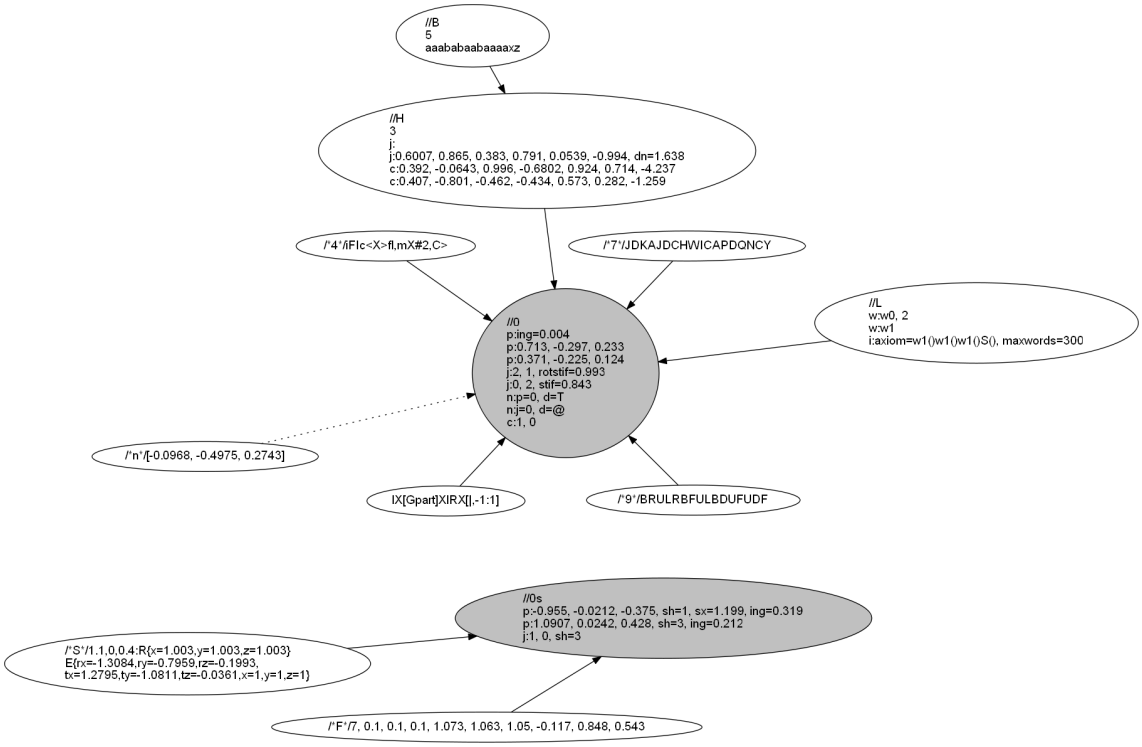


Figure 2.5: Genetic encodings in Framsticks and their example genotypes [126]. The encoding name appears in the first line of each genotype.

- **Joint** (`j:`)
  - part **indices** (`p1, p2`): indices of parts connected by the joint,
  - physical properties: **stiffness** (`stif`) and **rotation stiffness** (`rotstif`).
- **Neuron** (`n:`)
  - **index** of part (`p`) or joint (`j`) to which the neuron is attached,
  - **neuron description** (`d`) including **neuron class name** and optionally **neuron properties**.
- **Connection** (`c:`)
  - **indices** of the parent neuron (`n`) and source neuron (`i`),
  - **weight** of the connection (`w`).

The indices of parts, joints, and neurons correspond to line numbers, starting from 0. Fig. 2.6 illustrates an example of a genotype in the *f0* encoding and the resulting phenotype. This representation can accommodate any valid genotype and thus it encompasses the entire phenotype space. However, the *f0* is closer to a description of a model than a proper genetic encoding; it lacks modularity and information compression. Nevertheless, to enable its use as a genetic representation in evolutionary optimization, the genetic operators of mutation and crossover were defined.

```
//0
p:−1.582,−0.373,1.051
p:
p:1.0
j:0,1
j:1,2
n:j=0,d=|
```

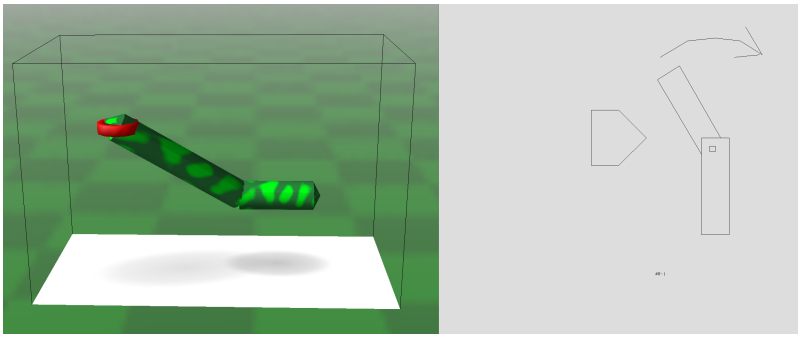


Figure 2.6: Example genotype in the *f0* encoding (left), a corresponding phenotype (middle), and its neural network (right).

Genetic operators

Mutation

The point mutation operator applies one of the following changes to a genotype:

- adding an object (a part, a joint, a neuron, or a neural connection),
- removing an object,
- modifying object's properties,
- swapping structure's parts.

Crossover

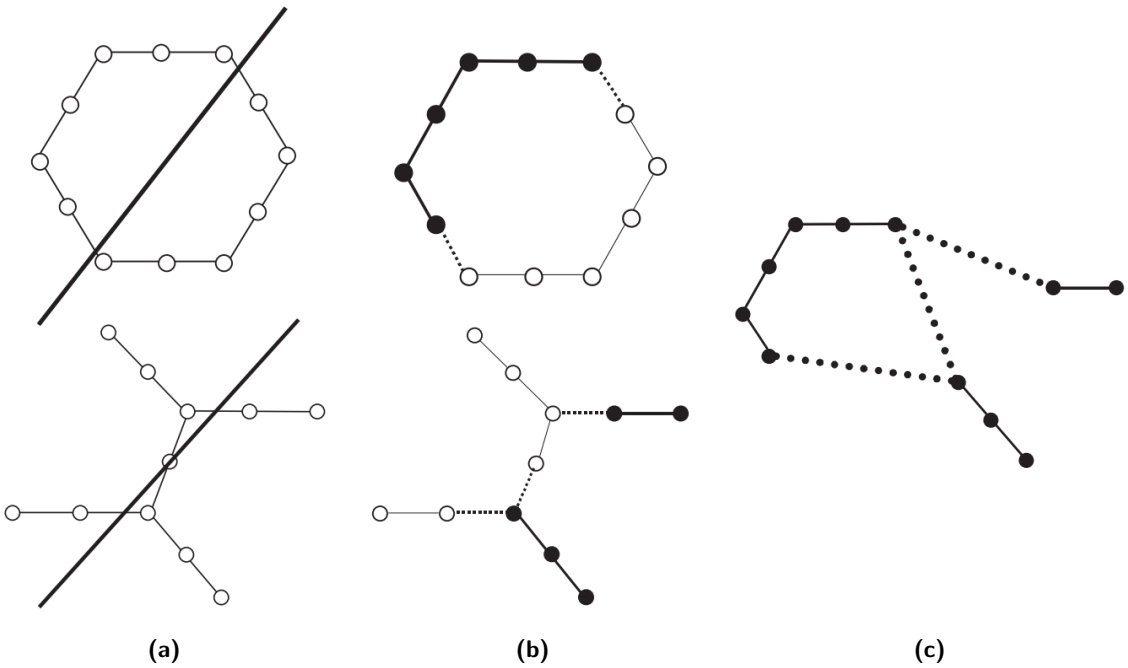


Figure 2.7: Crossover operator for the *f0* genetic encoding. 2.7a A randomly positioned plane divides the parent models into two parts. 2.7b The separated parts of the parents. 2.7c The child model formed by joining parts of the parent models. Dotted lines indicate newly created joints [123].

The development of the crossover operator posed more challenges to the authors as changing the line numbers in the genotype would change the reference numbers of the objects. Therefore, the crossover operator is based on the phenotypes. A randomly positioned plane is used to cut the parent phenotypes into two parts. Then the cut parts of the first and second parent are grafted together. Fig. 2.7 demonstrates the application of the crossover operation.

2.3.2 Recursive encoding (*f1*)

Syntax

The *f1* representation is a direct higher-level encoding. The connections between parts are implicit and do not require specifying parts indices. The resulting model is a tree-like structure with new sticks being connected to previous sticks. The basic symbols are:

- `X` – representing a stick (joint),
- `()` – representing a branch.

Within the parentheses, the commas are used to determine the angles between the branches. `X`s and the opening parenthesis (‘`(`’) can be preceded by modifiers that alter, depending on the modifier, either the following `X` or all the following `X`s. The modifications may affect stick position and its properties. Table 2.1 presents the list of modifiers available for the *f1* encoding.

In the *f1* encoding neurons are placed after the `X`s in square brackets, using the following syntax:

```
[NeuronType, PropertyAndInputList]
```

`PropertyAndInputList` is a list of pairs in the following format: `PropertyName:Value` and `NeuronInput:Weight`, delimited by comma. `NeuronInput` is a relative position of an input neuron in the genotype. A value of zero indicates the current neuron, whereas positive and negative values indicate the subsequent and antecedent neurons, respectively.

Fig. 2.8 illustrates an example of a genotype in the *f1* encoding and the corresponding phenotype.

`R`	rotation
`Q`	twist
`C`	curvedness
`L`	length
`F`	friction
`M`	muscle strength

Table 2.1: The modifiers employed in the *f1* encoding considered in this work. Upper and lower case letters respectively increase and decrease the value of the property [126].

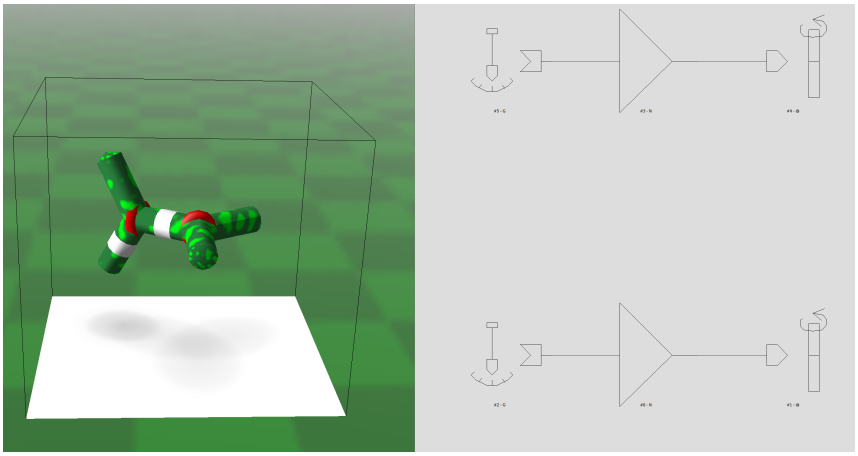


Figure 2.8: A phenotype (left) and its neural network (right) for a sample *f1* genotype: **X(X,**RRMM**X**[@G:.5]**(X**[@G:.5]**,X))**.

Genetic operators

Mutation

The mutation operator alters a genotype by performing one of the following operations:

- adding a modifier, stick, or neuron,
- deleting a modifier, stick, or neuron,
- modifying the parameter of a neuron.

Crossover

The crossover operator chooses a cutting points for each parent and swaps the resulting substrings. The cutting points are selected in a way that preserves the encoding syntax.

2.3.3 Developmental encoding (*f4*)

Syntax

The *f4* is an indirect developmental encoding that was inspired by the biological process of growth [123]. It employs similar symbols as the *f1* encoding, however with different interpretation. In the *f1* encoding, the symbols represent structural elements and their properties, while in the *f4* encoding, the symbols represent cellular actions. The development of a structure begins with a single undifferentiated cell. New cells are generated by executing instructions in parallel. The cells can either divide or differentiate into a stick (`X`) or a neuron (`N`). The development process terminates when there are no undifferentiated cells left.

The *f4* genotype is expressed as a string of characters, with most of the instructions consisting of a single character. Since each division creates two branches from the instruction sequence, the genotype can be represented as a binary tree. The additional codes that are included in the *f4* representation include `<`, `>`, and `#`. `<` denotes a division of a cell and the end of the current cell development is denoted by `>`. `#` allows for repetition of an

instruction, thus enabling the modularity of the *f4* representation. The full list of codes is presented in Table 2.2. Fig. 2.9 illustrates an example of a genotype in the *f4* encoding and the corresponding phenotype.

Genetic operators

Mutation

The mutation operator alters a genotype in its tree representation by performing one of the following operations:

1. adding a code,
2. deleting a code,
3. modifying a code,
4. modifying a parameter of a code.

Crossover

The crossover operator uses the genotype in its tree representation as well and works similarly to GP crossover operators, by swapping the subtrees of parent solutions.

`<`	division
`X`	turn into a stick
`N`	turn into a neuron, a receptor or a muscle
`>`	stop development of a cell
`,`	add a branch
`[...:...]`	add a neural connection
`#`	repetition marker

Table 2.2: The codes employed in the *f4* encoding [126]. This representation also uses modifiers, which are the same as for the *f1* representation (see Table 2.1).

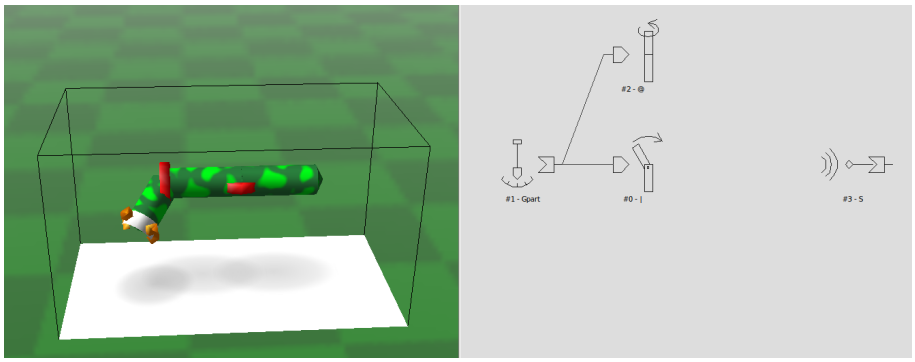


Figure 2.9: A phenotype (left) and its neural network (right) for a sample *f4* genotype: /*4*/<X><r,<X#2>><<XFl>c<N:@[−1:6.473]>N:S>N:Gpart> qmcN:|[1:2.298]#2>>.

2.4 3D structures data set

2.4.1 Data set creation

Population size	50
Generations	5000
Selection method	Tournament
Tournament size	5
Mutation probability	0.65
Crossover probability	0.25
Maximum number of body parts	30
Maximum number of body joints	30
Maximum number of neurons	20
Maximum number of neuronal connections	30

Table 2.3: Parameters and constraints of the evolutionary algorithm used to create a data set of genotypes of active and passive 3D structures.

The data set used for the analyses in the following chapters consists of $3 \times 4 \times 1000 = 12000$ genotypes: 1000 for each combination of 3 investigated genetic encodings (*f0*, *f1*, *f4*) and 4 considered optimization tasks. The genotypes comprising the set were obtained using a generational EA implemented in the Framsticks Python module `evolalg` [127]. The employed EA creates a new population by mutating, crossing over, and cloning a specified number of individuals in each iteration until a given number of generations is reached. The following optimization goals were considered:

- velocity on land,

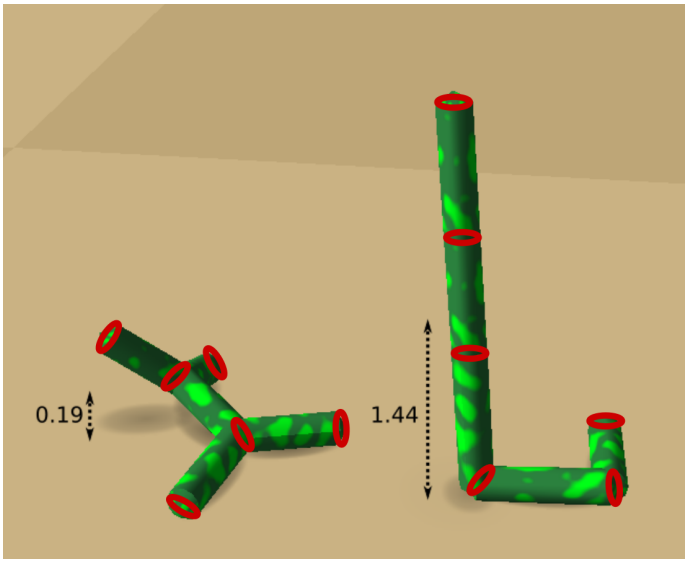


Figure 2.10: Two sample structures encoded using the *f1* encoding and their vertical position of the center of mass. The red ellipses correspond to the position of body parts, based on which the vertical position of the center of mass is calculated.

- velocity in water,
- height (a vertical position of the center of mass, see Fig. 2.10) of active structures (i.e. structures equipped with neural network),
- height (a vertical position of the center of mass) of passive structures (i.e. structures not equipped with neural network).

Each of the tasks involved maximizing the fitness function. The parameters of the evolutionary algorithm used to create the data set are presented in Table 2.3. These parameter values were determined based on the preliminary experiments to optimize the fitness of the solutions and ensure the convergence of the algorithm. To prevent unlimited growth of the complexity of the evolved structures, the following limits were applied:

- number of body parts: 30
- number of neurons: 20
- number of body joints: 30
- number of neural connections: 30

Any individual that exceeded any limit was deemed invalid and excluded from the population. A new individual was generated instead. When creating a new population, a tournament selection was used. First, selected 65% of individuals underwent mutation, then during crossing-over selected 25% pairs of individuals were replaced by their offspring, and finally selected 10% individuals were cloned.

In order to obtain a data set of diversified structures, the experiment was run 100 times for each combination of genetic encoding and optimization goal. In each run, the genotype of each new structure that had a higher fitness than the previous best was stored. The stored genotypes for each combination of genetic encoding and optimization goal were then sampled to obtain solutions that spanned the fitness range uniformly. The sampling procedure was as follows:

1. The maximum ($\texttt{fit}_{\max}$) and minimum ($\texttt{fit}_{\min}$) fitness values among the stored genotypes were computed.
2. The fitness value range was divided into 999 equal intervals. The lower bound of the i-th interval was given by:

$$x_i = \texttt{fit}_{\min} + i\frac{\texttt{fit}_{\max} - \texttt{fit}_{\min}}{n-1}, \quad i = 0, 1, ..., n-1$$

   where $n = 1000$
3. The stored genotypes were sorted in ascending order of their fitness values. Then, the following selection algorithm was applied, using the interval bounds as thresholds:

```
i = 0
for genotype in sorted_genotypes:
  if fitness(genotype) >= intervals_bounds[i]:
    selected.append(genotype)
    i += 1
```

This procedure resulted in a set of 1000 structures for each combination of genetic encoding and optimization objective, with their fitness values approximating an even distribution along the fitness range. Moreover, a smaller version of this set, containing 100 genotypes for each combination, was created using the same procedure. The larger set will be referred to as the `1000_data_set` and the smaller set as the `100_data_set`.

2.4.2 `1000_data_set` characteristics

Figs. B.1–B.4 in Appendix B show representative structures for each combination of genetic encoding and optimization goal. The shown structures were selected by applying the k-medoids algorithm [108] using the `opt` dissimilarity measure (described in Sect. 4.2.3) to evaluate the phenotypic distance. It can be seen that the solutions share the most phenotypic similarities between the representations for the *velocity on land* (Fig. B.1) task (albeit they can differ regarding the movement strategy). In this task, the morphologies for all representations tend to be smaller and less branched. The inter-representation differences are more pronounced for the *velocity in water* task, where the higher-level encodings *f1* and *f4* (especially *f1*) tended to produce more streamlined, elongated structures than the *f0* encoding (Fig. B.2). In the *height active* and *height passive* tasks (Figs. B.3–B.4) the evolved structures are, in general, greater in size than in two first tasks, as expected from the optimization objective. An interesting observation is that in the active version of the task, the structures tend to be more complex and branched than in the passive one. This suggests that in the presence of the neural network and effectors, the more complex morphology is more advantageous to maximize the vertical position.

Fig. 2.11 shows the fitness value distribution for each genetic encoding and each optimization task. The plots indicate that the sampling procedure generally achieved uniform fitness distribution, except for the *velocity in water* objective using the *f4* encoding, which suggests a more non-uniform fitness distribution for the developmental encoding in this task. The performance of each representation, measured by the maximum fitness value, varied across the optimization tasks. The only objective where the performance was comparable across the representations was the *height active* objective. Interestingly, in the *velocity in water* and *height passive* tasks, the low-level *f0* representation performed better than the recursive *f1* representation. This may imply that the more complex genotype-to-phenotype mapping of the *f1* representation creates a more rugged fitness landscape, which hinders the search in those tasks. The analysis of the fitness landscape for the considered representations and tasks will be the subject of Chapter 4.

Fig. 2.12 illustrates the body part distribution and Fig. 2.13 illustrates the bounding box size distribution of the structures. The number of parts and the bounding box size were lowest for the *velocity on land* objective (medians ranged from 4 to 5) and for the *f1* representation within this objective. In the *velocity in water* task, the medians (7-9) and the bounding boxes were higher. The higher-level encodings *f1* and *f4* produced larger structures in terms of both body parts and bounding boxes. The same was true for the *height active* objective. In the case of the *height passive* objective, *f1* and *f4* representations also produced larger structures in terms of bounding box size, but structures created using *f1* had generally fewer body parts than those created using *f4* and *f0*. This could be possible due to modifiers that enabled the elongation of body parts. It is noteworthy

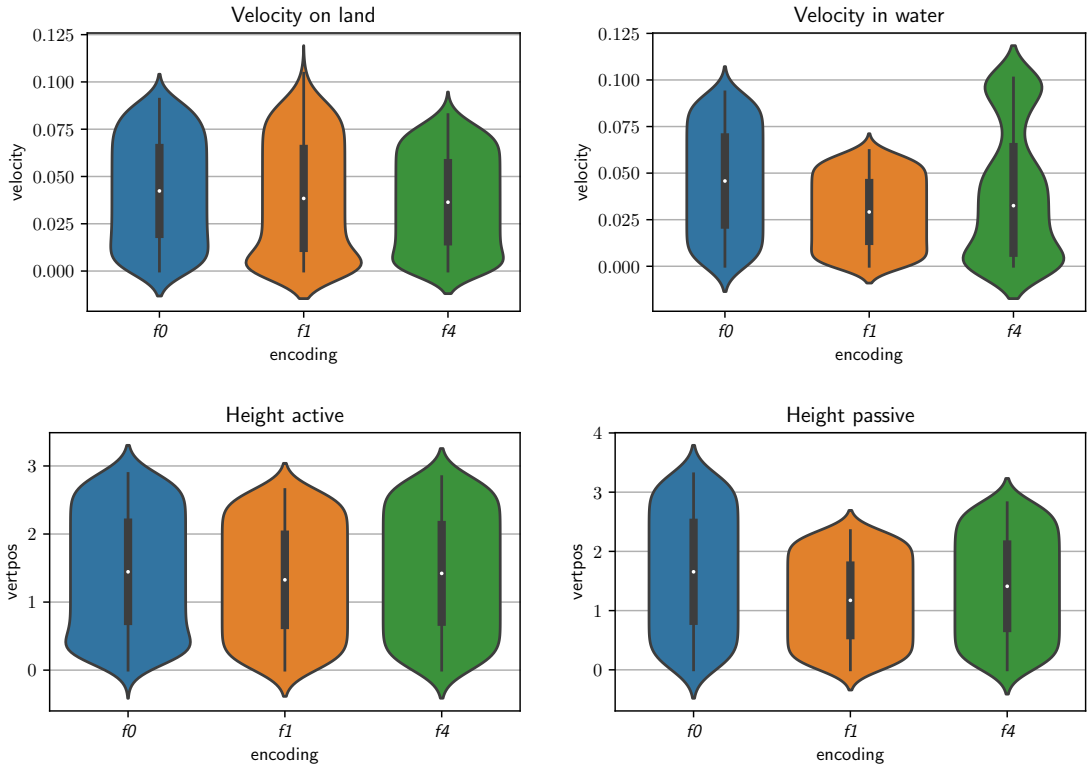


Figure 2.11: The distribution of fitness value for each of the four optimization tasks and each genetic encoding.

that in this objective, despite having the smallest bounding boxes, the structures created using the direct representation *f0* tended to achieve higher fitness. The visualization of the representative structures (Fig. B.4) shows that the *f0* allowed to create a "base" for supporting tall parts of a structure. The constraints of the other encodings resulted in more branched phenotypes.

Fig. 2.14 illustrates the neuron distribution in the optimization tasks involving active structures. In general, the structures created using the developmental encoding *f4* had fewer neurons. Except for the *velocity on land* objective, the direct encoding *f0* tended to produce structures with the most neurons. However, it should be noted that not all neurons present in a genotype necessarily have connections to other neurons or are effective.

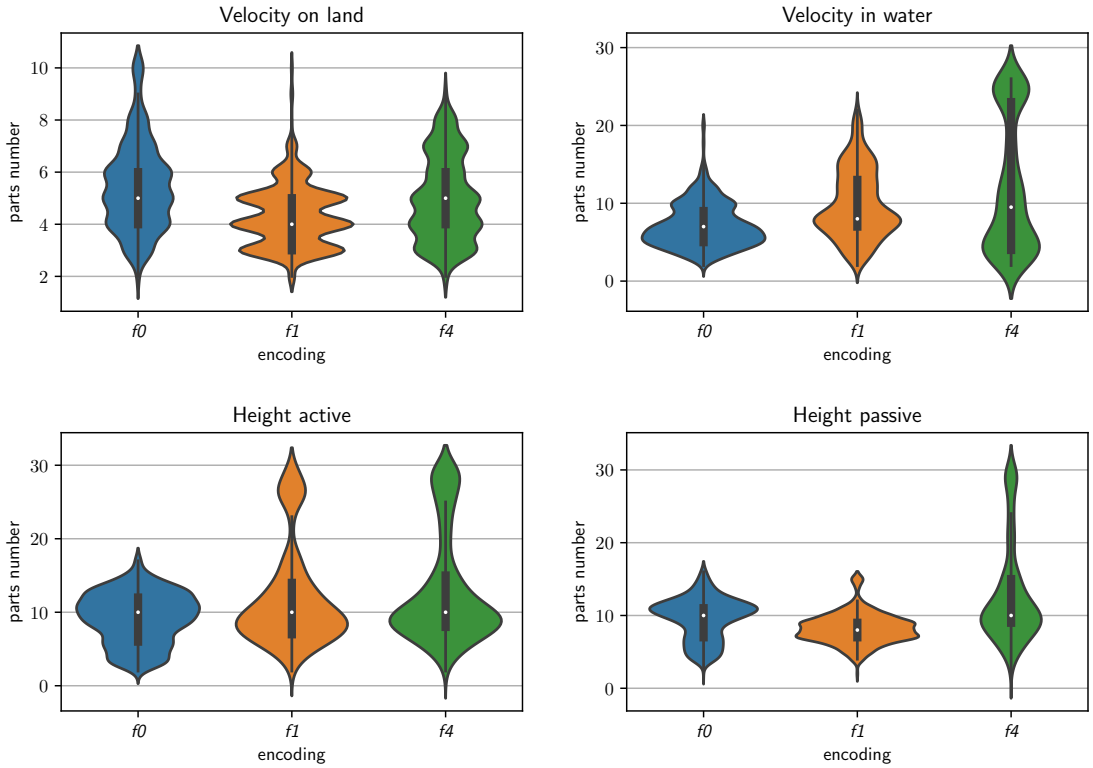


Figure 2.12: The distribution of body parts for each of the four optimization tasks and each genetic encoding.

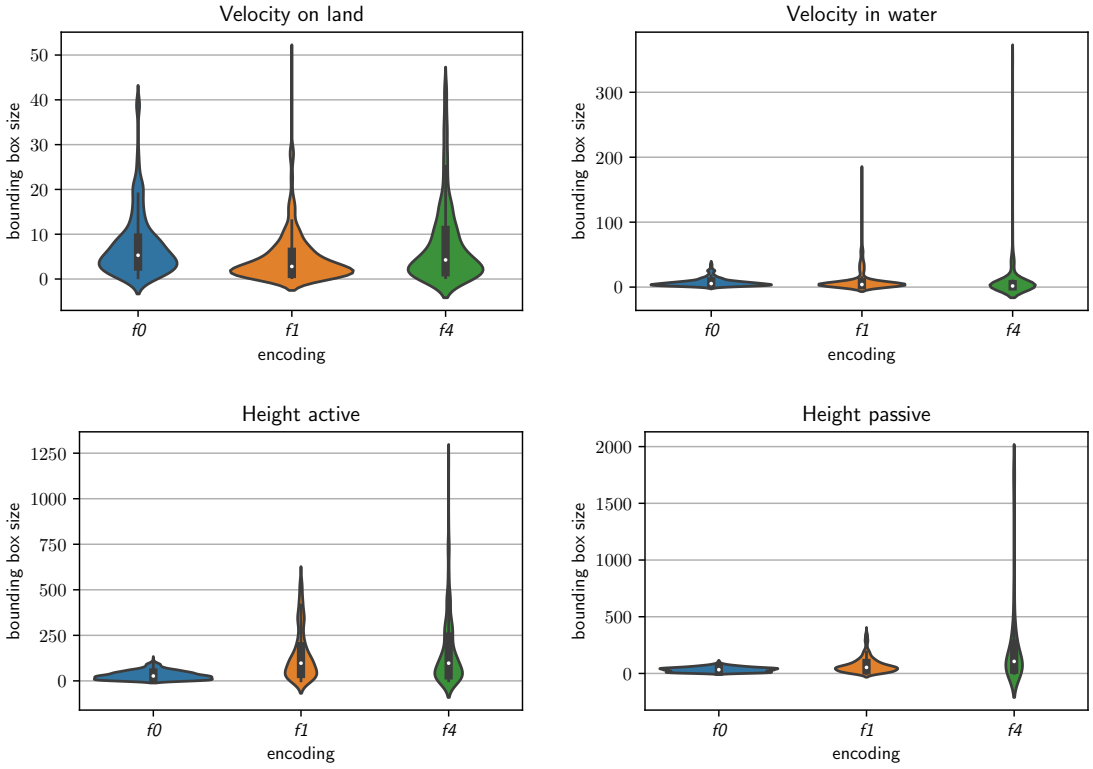


Figure 2.13: The distribution of bounding box size for each of the four optimization tasks and each genetic encoding (logarithmic scale). The bounding box size is the product of the x, y, and z dimensions of the smallest box that contains the structure.

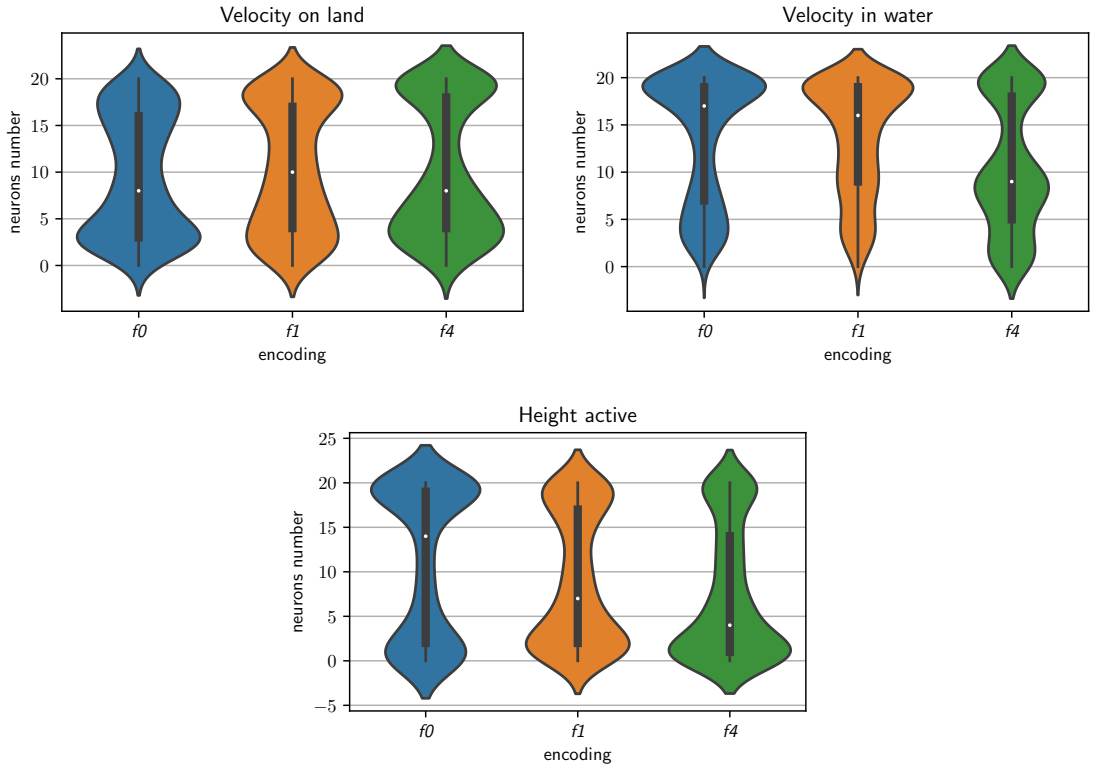


Figure 2.14: The distribution of neurons for each of the four optimization tasks and each genetic encodings.

2.5 Summary

This chapter introduced Framsticks, a versatile simulation software that will be employed in further analyses and computational experiments. Three different Framsticks genetic representations were selected for these experiments: a low-level encoding *f0*, a recursive encoding *f1*, and a developmental encoding *f4*. A data set of diverse 3D structures was generated using these representations and four distinct optimization objectives. The analysis of the data set showed that the structures evolved using different genetic representations had different fitness value distributions for the considered optimization objectives. They also differed in their size, complexity of morphology, and their neural network size. This diversified data set, the `1000_data_set`, and its smaller variant, the `100_data_set`, will be used in subsequent chapters for the analysis of the performance of dissimilarity measures, fitness landscape analysis, and the investigation of the properties of genetic operators for ED.

3

Dissimilarity Measures for 3D Structures

3.1 Introduction

Dissimilarity measures have multiple applications in evolutionary algorithms, including finding clusters of similar solutions [116], maintaining diversity [190], and designing genetic operators [61]. A set of established dissimilarity measures exists for the class of problems where the genotype is identical to the phenotype and a standard genetic representation (e.g. a bit string or a string of characters) is employed.

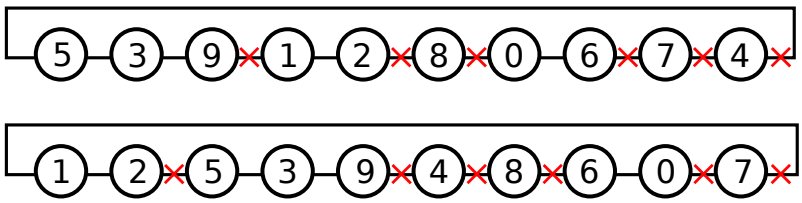
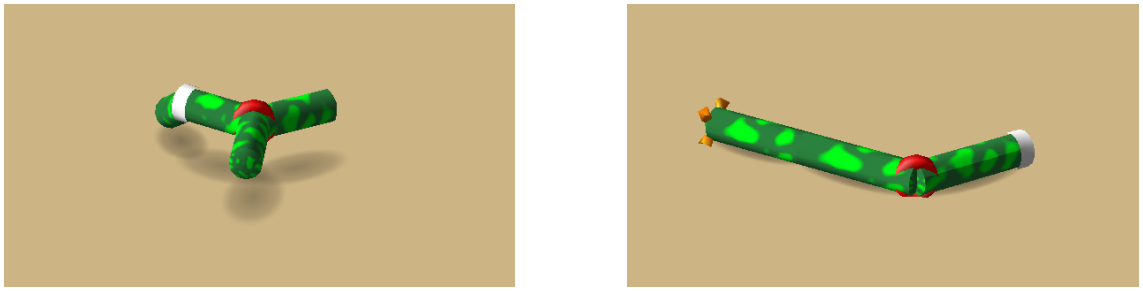


Figure 3.1: Two examples of solutions for the traveling salesman problem (TSP). The edges that differ between the solutions are indicated by red crosses. The dissimilarity measure is the number of distinct edges between the solutions, which is 6 for this pair [61].

Let us consider the traveling salesman problem (TSP) [105]. It is a combinatorial optimization problem that can be formulated as follows: given n cities that the salesman must visit and a cost of traveling between any two cities, determine a minimum cost route that visits each city exactly once and begins and ends in specified cities. A common representation for this problem is a permutation representation, where a solution is a sequence of integers that determines the order of visiting the cities. For this representation Freisleben and Merz proposed a distance preserving crossover (DPX) [61]. This genetic operator aims to produce an offspring solution distance of which to both parent solutions is equal to the distance between parents. The distance measure used in [61] is the number of edges (pairs of consecutive cities on the route) that they do not share. Fig. 3.1 shows two sample solutions for TSP represented as permutations and their distance value based on common edges.

Now let us consider two sample solutions evolved to maximize velocity on land, using



(a) X(QmX,QmfX[Gpart][@,−1:2.362,p:1]cX) **(b)** L(X[Gpart],fX[@,−1:1,p:1]X[S])

Figure 3.2: Two active structures and their genetic representation in the *f0* genetic encoding.

the *f1* genetic encoding (Fig 3.2). In this case, there are many more possibilities to quantify the dissimilarity of the solutions, as it can be assessed on a genotypic, phenotypic, or behavioral level. The genotype of the *f1* encoding consists of a string of characters, for which various distance metrics exist, such as Hamming distance [75], Levenshtein distance [142], and Jaccard distance [95], among others. However, relying on the genotype distance may not be the most suitable approach, because, as discussed in Sect. 1.3, low-locality representations may not preserve the correspondence between the genotype and the phenotype distances, and the phenotype is the subject of evaluation during the course of evolution.

On the other hand, assessing the dissimilarity between the phenotypes is a non-trivial, if not ill-posed, problem; there exists no ground truth and, in most cases, it is not possible to determine maximum dissimilarity. Of course, the problem of comparing 3D structures is present in various domains such as computer vision [94], chemical informatics [154], and bioinformatics [52]. Nevertheless, the measures used in those fields are often either too domain-specific or too computationally expensive to handle generic 3D structures of arbitrary complexity [118] (for instance, in chemical and bioinformatics models the chemical properties of the molecules may be also considered while computing dissimilarity). Hence, as a part of this work, a set of dissimilarity measures for 3D structures was devised. In order to be applied to a broad range of ED problems, such measures should exhibit properties that can be formulated as follows:

- **generality**: the measure should be applicable to any ED problem that can use a general representation, such as an undirected graph, regardless of the domain or the objective function,
- **discriminability**: the measure should be able to distinguish between morphologically different structures (possibly considering dissimilarity in neural networks),
- **extendability**: the measure should be able to incorporate additional information relevant to a specific domain into the dissimilarity assessment,
- **computational efficiency**: the measure should be able to handle 3D structures of arbitrary complexity without excessive computational cost.

This chapter introduces the set of measures developed or adapted with the aim of satisfying the aforementioned properties. The motivation for extending the set of measures

was to test how different aspects of phenetic dissimilarity, captured by different measures, would correspond to the difference in fitness value in evolutionary design problems.

The rest of this chapter is structured as follows. Sects. 3.2-3.6 present the proposed measures. Sect. 3.8 compares the measures with regard to the computational efficiency, in terms of the correlation between dissimilarity values for a set of structures, and using a qualitative comparison. Sect. 3.9 presents the results of the investigation of human perception of similarity. Sect. 3.10 summarizes the main findings.

3.2 The Levenshtein distance (gene)

Levenshtein distance [142] is used here to compute the dissimilarity between genotypes. It is a dissimilarity metric for strings defined as a minimum number of single-character edits required to transform one string into another. The single-character edits include:

- inserting a character into the string,
- deleting a character from the string,
- substituting a character.

Formally, the Levenshtein distance $\mathrm{lev}(a, b)$ between strings a and b of lengths $|a|$ and $|b|$, respectively, is defined as:

$$\mathrm{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \mathrm{lev}\left(\mathrm{tail}(a), \mathrm{tail}(b)\right) & \text{if } a[0] = b[0], \\ 1 + \min \begin{cases} \mathrm{lev}\left(\mathrm{tail}(a), b\right) \\ \mathrm{lev}\left(a, \mathrm{tail}(b)\right) \\ \mathrm{lev}\left(\mathrm{tail}(a), \mathrm{tail}(b)\right) \end{cases} & \text{otherwise} \end{cases}$$

Let us consider two genotypes in the *f1* encoding:

1. L**(X,**f**XX)**
2. **X**Q**m****X**

The sequence needed to transform the first string into another is as follows:

1. Delete “L” from the beginning of the first string.
2. Delete “(” from the second position of the first string.
3. Substitute “,” at the third position of the first string with “Q”.
4. Substitute “f” at the third position of the first string with “m”.
5. Delete “X” from the seventh position of the first string.
6. Delete “)” from the eights position of the first string.

Hence the Levenshtein distance between these two genotypes is equal to 6.

It is important to note that any digits in a genotype, such as neural connection weights or references to part numbers in the *f0* encoding, are treated the same as any other character. This means that the information about numerical components is not utilized as effectively as it could be if the dissimilarity between numbers was handled separately.

This measure will be referred to as `gene`. The goal of including a simple genotype-based measure is to provide a reference for the phenetic measures.

3.3 The heuristic graph-based measure (greedy)

3.3.1 Assumptions

The first of the analyzed phenetic measures was introduced by Komosinski and Kubiak [118] and implemented natively in Framsticks. The measure models 3D structures as undirected graphs where body parts correspond to vertices and body joints correspond to edges. Each vertex in a structure can be described with the following properties:

- its degree (i.e. the number of edges incident to the vertex),
- the number of neurons attached to the vertex,
- its position in a three-dimensional coordinate system.

Fig. 3.3 illustrates an example of a structure and its corresponding graph-based model used by the measure.

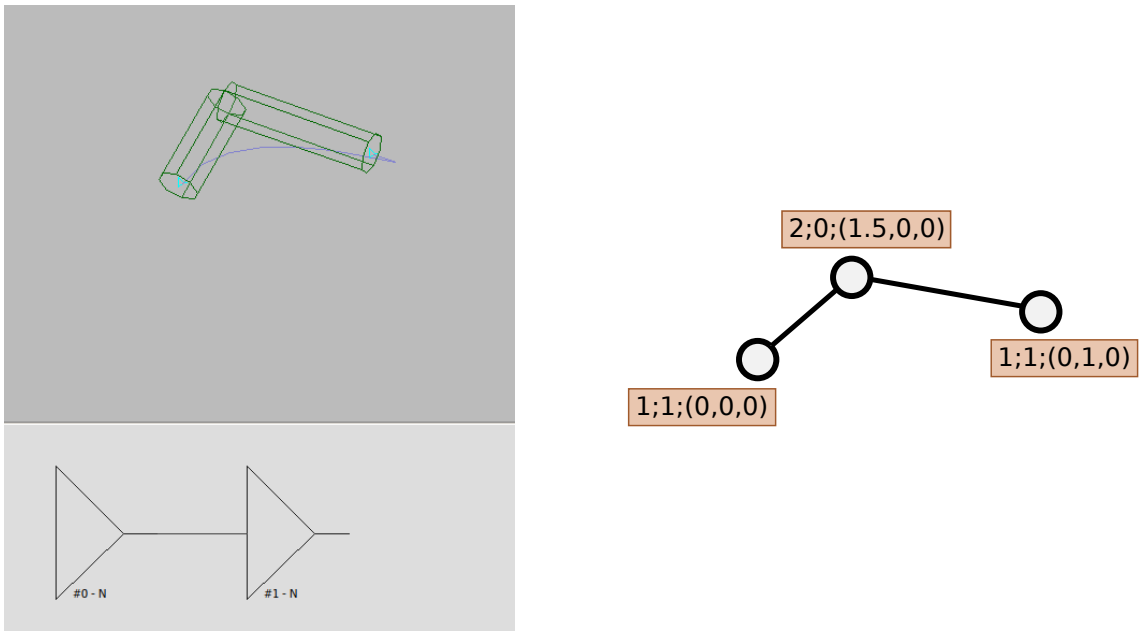


Figure 3.3: A simple three-part structure with two neurons, each attached to a different body part: body and brain (left column), conceptual model employed in `greedy` and `opt` measures (right column). Numbers in the rectangles denote vertex parameters: vertex degree, number of neurons attached to the vertex, and coordinates in a three-dimensional coordinate system. The structure was created from the following genotype: **(,**LF**X**[N]**,X**[N,−1:1]**)**

The measure employs a heuristic algorithm that aims to find the most similar pairs of vertices from the two structures and compute the structures' dissimilarity based on the matching. The value of the dissimilarity consists of four components:

- $\mathtt{d_V}$ – the absolute difference in the number of vertices in both structures,
- $\mathtt{d_D}$ – the sum of absolute difference in the degree of matched vertices,
- $\mathtt{d_N}$ – the sum of absolute difference in the number of neurons attached to the matched vertices,
- $\mathtt{d_G}$ – the sum of Euclidean distance between matched vertices.

The values of those components are aggregated using the weighted sum:

$$dissim = \mathtt{w_V} \cdot \mathtt{d_V} + \mathtt{w_D} \cdot \mathtt{d_D} + \mathtt{w_N} \cdot \mathtt{d_N} + \mathtt{w_G} \cdot \mathtt{d_G}$$

The user can adjust the importance of each component by setting the weight ($\mathtt{w_V}$, $\mathtt{w_D}$, $\mathtt{w_N}$ and $\mathtt{w_G}$) of this component to a value greater than or equal to zero.

The algorithm of the `greedy` measure consists of three main steps:

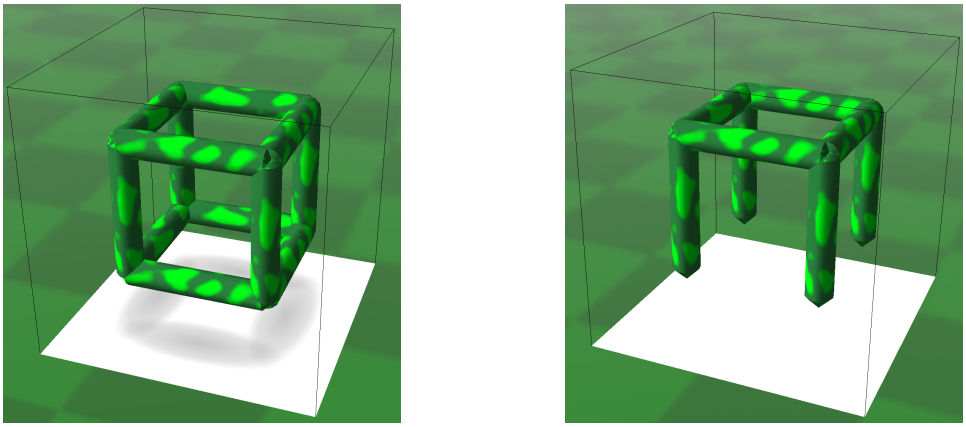
1. Aligning the structures,
2. Matching the vertices,
3. Calculating the dissimilarity,

which are described in more detail in the following sections.

3.3.2 Alignment procedure

The first step, the alignment of the structures, is performed only when the $\mathtt{w_G}$ weight is greater than zero. The aim of this procedure is to find the best spatial alignment of the two structures. For this purpose, in the original implementation of the measure, a multidimensional scaling [44] procedure (MDS) has been used for each structure separately. The MDS algorithm takes the distance matrix between vertices as input. The new coordinates are set based on the three dimensions with the highest variance. After the application of this procedure, the centers of both structures are located at the origin of the coordinate system. The axis with the highest variance of coordinates is chosen as the first axis of the structure, and the axis with the second highest variance of coordinates is chosen as the second axis of the structure.

However, this procedure only considered the vertices of the structures, without taking into account the edges. As a result, vertices with a similar vertex degree in both structures might not have been aligned correctly. Fig. 3.4 illustrates an example of two structures that were indistinguishable for the alignment procedure: a "cube" (Fig. 3.4a) and a "table" (Fig. 3.4b). Moreover, the vertices of the "table" structure and its rotated versions could have been misaligned by the original alignment procedure. To overcome this problem, the information about the vertex degree had to be incorporated into the alignment process. To achieve this goal, as a part of this work, the MDS algorithm was replaced with weighted classical MDS [119], using vertex degrees as weights for double centering the distance matrix.



(a) "Cube" **(b)** "Table"

Figure 3.4: Two structures sharing the coordinates of all the vertices and differing in the number of edges connecting the vertices.

3.3.3 The matching procedure

The main part of the algorithm is the construction of the matching function. In order to build the matching, the vertices of each structure are sorted according to the vertex degree, and then according to the number of neurons within the groups of the same vertex degree. The matching procedure starts with vertices with the highest vertex degree in both structures and tries to find the pairs of matching vertices. The within-group matching ends when there are no unmatched vertices with a given degree in one or both of the structures. Then the algorithm moves to the next group of vertices with the second highest degree, and continues until all groups are processed. Within each group, the algorithm matches vertices based on the minimum distance calculated for each pair of vertices as:

$$dist_{ij} = \mathsf{w_D} \cdot \mathsf{d_D}(v_{i1}, v_{j2}) + \mathsf{w_N} \cdot \mathsf{d_N}(v_{i1}, v_{j2}) + \mathsf{w_G} \cdot \mathsf{d_G}(v_{i1}, v_{j2}) \tag{3.1}$$

where v_{i1} denotes the i-th vertex of the first structure, and v_{j2} denotes the j-th vertex of the second structure. Listing 3.1 details the matching procedure.

```
1  sort(S1, descending by vertex degree then neuron count)
2  sort(S2, descending by vertex degree then neuron count)
3  while !(all vertices in smaller(S1, S2) are matched):
4    FIND U, V - groups of vertices in S1, S2 with the
5    highest possible degree and with unmatched parts
6    while !(all u ∈ U or v ∈ V are matched):
7        FIND the first vertex u ∈ U which is yet unmatched
8        FIND the first vertex v ∈ V which is yet unmatched
9        min_for_u = min_dist(u,V)
10       min_for_v = min_dist(v,U)
11       match(u, min_for_u)
12       match(v, min_for_v)
```

Listing 3.1: The matching procedure of the greedy measure. S_1 and S_2 denote part sets of the two structures being matched.

3.3.4 Dissimilarity calculation

After matching the vertices, the dissimilarity is calculated as the sum of distances between the matched vertices using the formula 3.1. When the structures have different numbers of vertices, some vertices of the larger structure will not have a match in the smaller structure. These unmatched vertices contribute to the penalty term added to the dissimilarity measure formula. The penalty for each unmatched vertex v_i is calculated as follows:

- $\text{penalty}_D(v_i) = \mathtt{w_D} \cdot \text{vertex\_degree}(v_i)$,
- $\text{penalty}_N(v_i) = \mathtt{w_N} \cdot \text{number\_of\_neurons}(v_i)$,
- $\text{penalty}_G(v_i) = \mathtt{w_G} \cdot \text{distance\_to\_the\_origin}(v_i)$,

where $\text{distance\_to\_the\_origin}(v_i)$ is the Euclidean distance of the vertex v_i to the origin of the coordinate system. The penalties for all unmatched vertices are summed up and added to the distance between the matched vertices. The dissimilarity value also includes the difference in the number of unattached neurons (neurons that are not attached to any body part) and in the number of vertices of the structures, weighted by $\mathtt{w_N}$ and $\mathtt{w_V}$, respectively. Listing 3.2 shows the pseudocode of the final dissimilarity value calculation.

```
distance = 0
# add the weighted distance between matched vertices
for (u, v) in matched_vertices_pairs:
  distance += w_D · d_D(u, v) + w_N · d_N(u, v) + w_G · d_G(u, v)
# add the penalty for the unmatched vertices
for u in unmatched_vertices:
  distance += penalty_D(u) + penalty_N(u) + penalty_G(u)
# add the weighted difference in unattached neurons
distance += w_N · difference_in_unattached_neurons(S_1, S_2)
# add the weighted difference in the number of vertices
distance += w_V · d_V(S_1, S_2)
```

Listing 3.2: The distance calcuation in the `greedy` measure. S_1 and S_2 denote part sets of the two structures being matched.

3.3.5 Summary

The `greedy` measure is a fast heuristic algorithm that computes dissimilarity based on the graph model of the structures, incorporating information about vertex degree, neuron count, and geometric distance. However, this measure has some limitations. One of them is the fixed order of vertex matching, which starts from the vertices with the highest degree and follows a predefined logic, regardless of the weight of the $\mathtt{d_D}$ component. Another drawback is the greedy nature of the matching algorithm. It always chooses the matching which provides the minimal distance between currently considered vertices, however, this choice does not have to result in the minimal total distance between the structures. To address these challenges, as a part of this work, the `opt` measure was developed.

3.4 The optimal-matching graph-based measure (opt)

3.4.1 Differences compared to the greedy measure

	V_{S1}	V_{S2}	V_{S3}	V_{S4}	V_{S5}	$\mathbf{P_1}$
V_{G1}	2.27	1.45	2.01	2.76	7.67	2.32
V_{G2}	1.68	3.53	1.85	2.18	8.98	3.50
V_{G3}	1.47	1.24	2.84	3.00	7.49	2.32
V_{G4}	3.60	5.53	2.85	2.30	7.01	4.50
V_{G5}	4.26	4.15	3.49	2.97	4.97	3.32
V_{G6}	3.26	3.15	2.49	1.97	5.53	2.32

Table 3.1: A distance matrix for two structures with different numbers of parts. The rows correspond to the parts of the larger structure, which has 6 parts, and the columns correspond to the parts of the smaller structure, which has 5 parts. The column $\mathbf{P_1}$ shows the penalty for each part of the larger structure assigned in the case it is not matched with a vertex from the smaller structure. The penalty is calculated in the same way as in the greedy measure (Sect. 3.3.4).

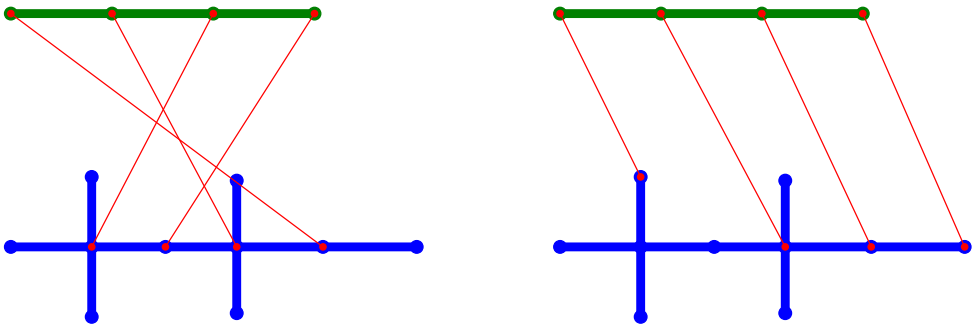
The `opt` measure is the first measure developed as a part of this work. It is an improved version of the `greedy` measure. It has the same parameters and the same three main steps: aligning the compared structures, constructing the matching function, and calculating the dissimilarity. The key difference is the algorithm used for constructing the matching function. Instead of using a greedy procedure, the `opt` measure employs the Kuhn-Munkres algorithm [133, 158] (also known as the Hungarian algorithm), which finds the optimal matching of vertices that minimizes the total distance between two structures. The distance function is the same as in the `greedy` measure (3.1). This approach obviates the need for sorting the vertices by their degree. Furthermore, in the `opt` measure all components are treated uniformly, which facilitates the customization of the measure by incorporating additional vertex properties as subsequent components.

Comparison of matching of sample structures

Fig. 3.5 shows the vertex matching for two structures using the `greedy` and `opt` measures. The greedy matching procedure, starting from the vertices with the highest vertex degree in both structures, results in a suboptimal matching and a higher dissimilarity value (Fig. 3.5a). The application of the Kuhn-Munkres algorithm in the `opt` measure for the matching procedure allows for minimizing the dissimilarity value (Fig. 3.5b).

3.4.2 Distance calculation

```
# "greater" structure
GS = structure with more vertices
# "smaller" structure
SS = structure with less vertices
```



(a) greedy= 51.55 **(b)** opt= 42.58

Figure 3.5: A comparison of two structures and their vertex matching using the greedy and opt measures. The red lines indicate the matched vertices between the structures. The dissimilarity values for both measures are shown below the structures [119].

```
nGreat, nSmall = vertices_number(GS), vertices_number(SS)
dist_matrix = matrix(size=(nGreat, nGreat))
# fill the distance matrix
for i in range(0, nGreat-1):
  for j in range(0, nGreat-1):
    if j >= nSmall:
      dist_matrix[i][j] = penalty(v_GSi)
    else:
      dist_matrix[i][j] = dist(v_SSi,v_GSj)
# match the vertices using the Kuhn-Munkres algorithm
matched = HungarianAlgorithm(dist_matrix)
# calculate the distance between matched vertices
distance = sum(dist(matched))
# add the weighted difference in unattached neurons
distance += difference_in_unattached_neurons(GS, SS)
# add the weighted difference in the number of vertices
distance += w_V · d_V(GS, SS)
```

Listing 3.3: The algorithm for vertex matching procedure and distance calculation of the opt measure.

The dissimilarity value is calculated similarly as in the greedy measure and consists of the following components: the sum of the distances between matched vertices, the penalty for the unmatched vertices (only present when the structures have different numbers of vertices), the difference in unattached neurons, and the difference in the number of vertices. Unlike in the greedy algorithm, the penalty for the unmatched vertices is taken into account during the matching procedure. To do this, the vertices distance matrix is extended with additional rows or columns. For each vertex of the larger structure, these extra rows or columns are filled with the penalty value for that vertex for not having a match in the smaller structure (calculated as described in Sect. 3.3.4). Hence, matching a vertex from the larger structure to a vertex from the surrogate row or column corresponds to adding the penalty for being unmatched to the overall distance. Table 3.1 illustrates

an example of a distance matrix for two structures with different numbers of vertices. Listing 3.3 shows the pseudocode of the matching procedure and distance calculation for the `opt` measure.

3.5 The descriptors-based measure (shape)

3.5.1 Introduction

The `shape` measure developed in this work is based on the method proposed by Osada et al. [166] for matching 3D models. The algorithm involves creating histograms of simple shape descriptors, separately for each structure, and computing the distance between the shape descriptor distributions. This approach to evaluating dissimilarity of 3D structures offers several advantages. First, it eliminates the need for aligning the structures, which is a challenging task in itself and can significantly affect the resulting dissimilarity value. Second, this approach enables parallelization, as shape descriptor distributions are calculated separately for each structure.

3.5.2 The algorithm

The algorithm can be divided into four main steps. The first three are applied to each structure separately. First, the surface of the structure is sampled uniformly with a given density. Second, a specified number of point pairs, triplets or quadruplets (depending on the chosen descriptor function) are randomly selected from the sampled points. Third, the descriptor values are calculated and histograms with fixed bin sizes are constructed and normalized by dividing by the histogram total. Finally, the distance between the histograms is measured using the Earth Mover's Distance (EMD) [188].

The `shape` measure has five parameters which are listed below:

- `descriptor` – name of a descriptor function,
- `sampling density` – a parameter that controls the resolution of the surface sampling process: for each unit square of the bounding box of the 3D model, at least (`sampling density`)^2 points are sampled from the surface,
- `samples number` – the number of points that are randomly selected from the sampled surface and passed to the descriptor function,
- `bins number` – the number of bins that are used to construct a histogram from the values obtained by applying the descriptor function to the selected points.

Five descriptor functions were implemented, following Osada et al. [166]:

- `a3` – an angle of a triangle whose vertices are three randomly sampled points from the surface of the structure
- `d1` – the distance from the centroid of the structure to a randomly sampled point on its surface
- `d2` – the distance between two randomly sampled points on the surface of the structure

- `d3` – the square root of the area of a triangle whose vertices are three randomly sampled points from the surface of the structure
- `d4` – the cube root of the volume of a tetrahedron whose vertices are four randomly sampled points from the surface of the structure

Listing 3.4 outlines the algorithm for the `shape` measure.

```
# Input: two 3D structures S1 and S2, a descriptor function name, a number of samples N, a sampling density D, and a number of histogram bins B
# Output: a dissimilarity value between S1 and S2

# Step 1: Sample the surface of each structure with density D
# returns a set of points on the surface of S1
S1_samples = sample_surface(S1, D)
# returns a set of points on the surface of S2
S2_samples = sample_surface(S2, D)

# Step 2: Randomly select N point pairs, triplets or quadruplets from each sampled surface
# returns a list of tuples of points from S1_samples
S1_points = select_points(S1_samples, N)
# returns a list of tuples of points from S2_samples
S2_points = select_points(S2_samples, N)

# Step 3: Compute the descriptor value for each tuple of points using the descriptor function and construct normalized histograms with B bins for each structure
# returns a list of scalar values for each tuple in S1_points
S1_descriptor_vector = calculate_descriptor(S1_points, descriptor)
# returns a list of scalar values for each tuple in S2_points
S2_descriptor_vector = calculate_descriptor(S2_points, descriptor)

# returns a list of frequencies for each bin
S1_histogram = construct_histogram(S1_descriptor_vector, B)
# returns a list of frequencies for each bin
S2_histogram = construct_histogram(S2_descriptor_vector, B)

# returns a list of probabilities for each bin
S1_histogram_normalized = normalize_histogram(S1_histogram)
# returns a list of probabilities for each bin
S2_histogram_normalized = normalize_histogram(S2_histogram)

# Step 4: Calculate the Earth Mover's Distance (EMD) between the normalized histograms as the dissimilarity value
# returns a scalar value
dissimilarity = calculate_EMD(S1_histogram_normalized, S2_histogram_normalized)
```

```
35
36 return dissimilarity
```

Listing 3.4: The algorithm for the shape measure computation.

3.5.3 Parameter tuning

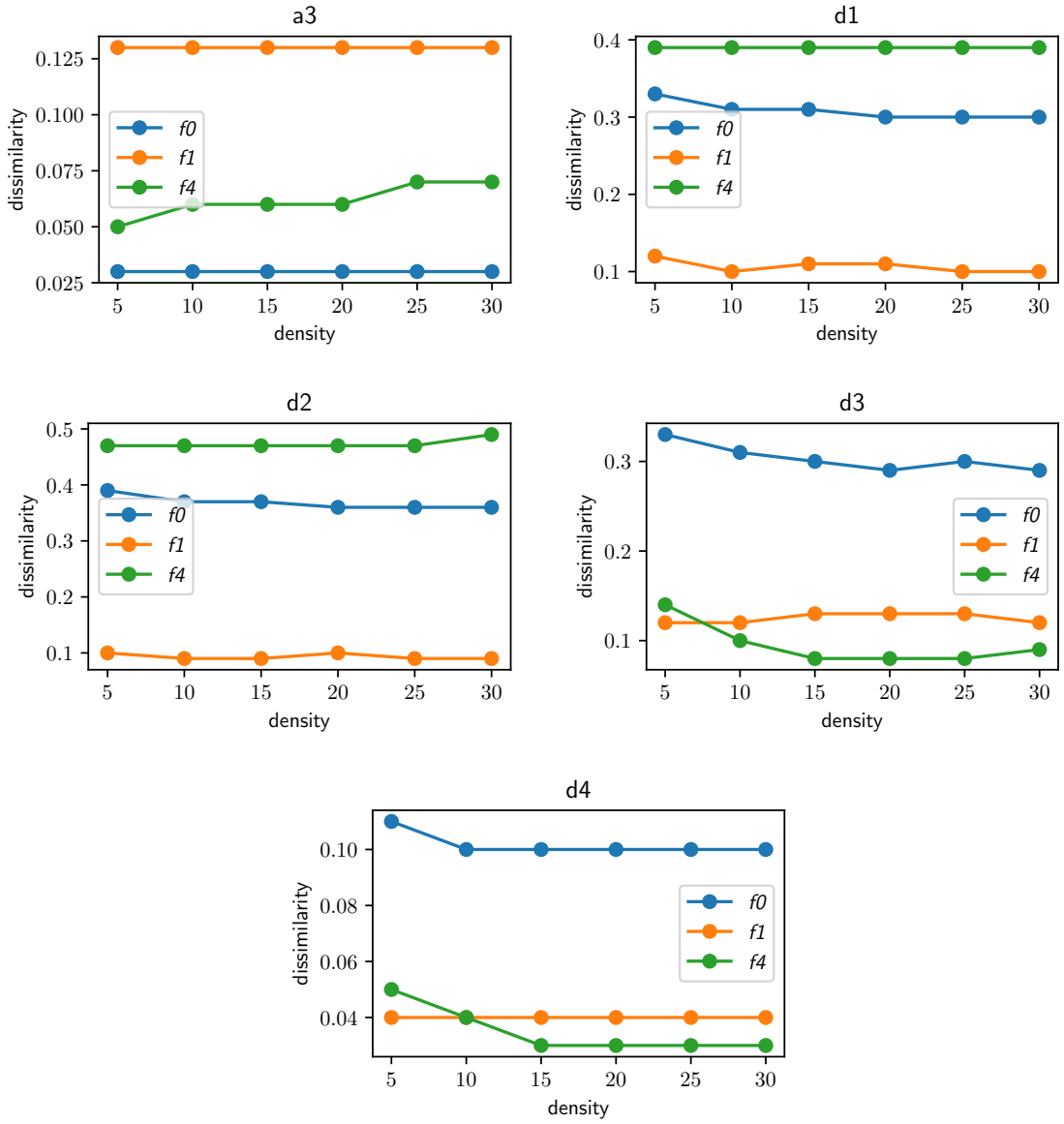


Figure 3.6: Dissimilarity value for three sample pairs of structures encoded using *f0*, *f1*, and *f4* and increasing value of the `density` parameter. The bin number was set to 1024 and the samples number to 100 000.

The parameter values of the `shape` measure, used in the analyses presented in the subsequent chapters, were determined experimentally. Since there is no ground truth for comparing the dissimilarity value, the parameter tuning was based on the observation that

the dissimilarity value tends to stabilize as the parameter values increase, indicating that further increasing of the parameter values will not improve the precision of the result. Therefore, the aim of the following experiments was to identify the threshold values of the parameters beyond which the dissimilarity value remains constant or nearly constant. The first step in the parameter tuning was to set the value of `density`, which controls the resolution of sampling of the 3D model surface, as this is the most computationally intensive step in the algorithm. The other parameter values were fixed as follows:

- `bins number`=1024,
- `samples number`=100 000,

to achieve high precision of dissimilarity values. Fig. 3.6 show the dissimilarity values computed for sample pairs of structures with different `density` values. The figure indicates that, in most cases, the dissimilarity values remain constant or vary only slightly for `density` values of 10 or higher. Therefore, in the subsequent experiments, the value of `density` was set to 10.

Fig. 3.7 shows the effect of varying `samples number` and `bins number` on the dissimilarity value calculated using descriptor `d1`, with a fixed `density` of 10. The other descriptors yielded similar results, which are reported in Appendix C. The results indicate that, in most cases, the dissimilarity value was stable or changed minimally for increasing `bins number`. Therefore, `bins number` was set to a low value of 128. On the other hand, as

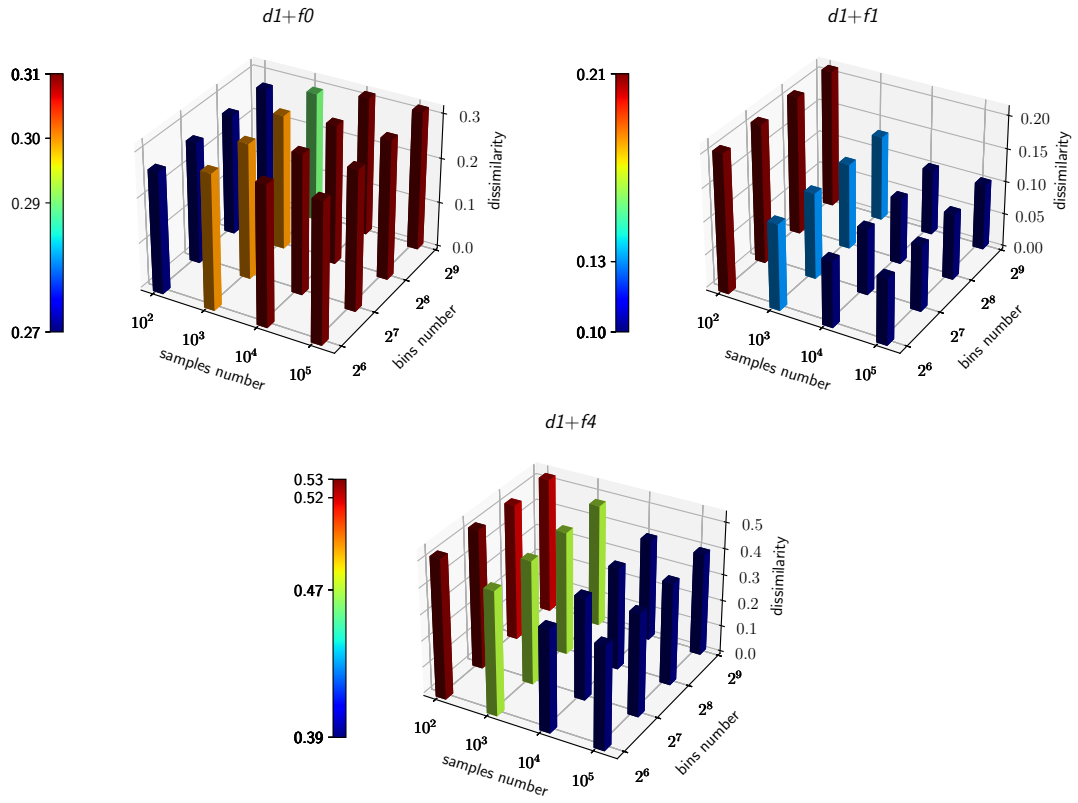


Figure 3.7: Dissimilarity value for various combinations of `samples number` and `bins number` using descriptor `d1`. The color of each bar indicates the dissimilarity value. The `density` parameter value was fixed at 10.

`samples number` increased, the dissimilarity tended to converge to a certain value. In most cases, this value was reached for `samples number` equal to 10 000, which was the value of the parameter that was selected.

3.6 The distribution-based measures (`dens` and `freq`)

3.6.1 Introduction

The last two of the measures developed in this work assess the dissimilarity of 3D structures based on their spatial distribution, using two approaches:

1. comparing the sampled surfaces of structures (`dens` measure),
2. comparing the frequency domain representations of sampled structures' surfaces (`freq` measure).

Both measures utilize a common algorithm to calculate the dissimilarity, except for the steps that involve the computation of the signature. Hence, they are described jointly in the subsequent section.

3.6.2 The algorithm

The algorithm consists of four main steps. First, the compared structures are spatially aligned using the weighted MDS procedure, which is described in Sect. 3.3. Second, the surface of each structure is uniformly sampled with a given density. Third, the bounding box of the aligned structures is partitioned with a specified resolution into cuboids. A signature, which serves as the basis for dissimilarity calculation, is computed based on the number of samples falling into each cuboid. For the `dens` measure, the signature is a vector of tuples, where each tuple corresponds to a subsequent cuboid and contains the centroid of the samples within the cuboid and the number of the samples. For the `freq` measure, the signature is obtained by applying Fast Fourier Transform (FFT) [25] to the vector of sample counts for each cuboid, and taking the absolute value of the resulting Fourier coefficients.

Since only the structure's surface is sampled, the number of empty cuboids may be significant, hence the algorithm provides the option to reduce the signature, i.e. remove the vector elements that have the count of samples equal to zero in both structures. For the `freq` measure, this reduction occurs prior to applying the FFT. Furthermore, for the `freq` measure, the indices of the samples count vector are scaled to the interval [0,1] to avoid the dependence of the dissimilarity value on the vector size.

Finally, the dissimilarity between the structures is computed by calculating the distance between their signatures according to the chosen distance metric.

The measure has four parameters which are listed below:

- `sampling density` – a parameter that controls the resolution of sampling the surface of the 3D solid shape, such that at least $(\texttt{sampling density})^2$ points are sampled from each unit square of the model's bounding box,

- `resolution` – determines the number of intervals used in each dimension to partition surface samples of structures in the 3D space (a higher value implies a more detailed comparison and a longer computation time),
- `reduce empty` – a parameter that indicates whether to remove or retain the elements in the signature vectors that correspond to cuboids that are empty for both structures (its default value is `True`),
- `metric` – a parameter that selects the distance metric to be used for calculating the dissimilarity (EMD, L1, or L2, with EMD as the default metric).

Listing 3.5 outlines the algorithm for the distribution based measures, `dens` and `freq`.

```
# Input: structures S1 and S2, sampling density D, resolution R, reduce empty
    flag reduce_empty, distance metric distance_metric
# Output: dissimilarity

# Step 1: Align structures using weighted MDS
# returns aligned structure S1
S1_aligned = weighted_MDS(S1)
# returns aligned structure S1
S2_aligned = weighted_MDS(S2)

# Step 2: Sample surface of structures with density D
# returns a set of points on the surface of S1
S1_samples = sample_surface(S1_aligned, D)
# returns a set of points on the surface of S2
S2_samples = sample_surface(S2_aligned, D)

# Step 3: Partition bounding box into cuboids with resolution R
# returns a list of cuboids that cover the bounding box
cuboids = partition_bounding_box(S1_aligned, S2_aligned, R)

# Step 4: Compute signature based on the number of samples in each cuboid
if measure == DENS:
  # Signature is a vector of tuples (centroid, count)
  S1_signature = []
  S2_signature = []
  for c in cuboids:
    # returns the number of samples from S1 in cuboid c
    S1_count = count_samples(S1_samples, c)
    # returns the number of samples from S2 in cuboid c
    S2_count = count_samples(S2_samples, c)
      if reduce_empty and S1_count == 0 and S2_count == 0:
      # Skip empty cuboid if reduce flag is set
      continue

```

```
    # returns the centroid of the samples from S1 in cuboid c
    S1_centroid = compute_centroid(S1_samples, c)
    # returns the centroid of the samples from S2 in cuboid c
    S2_centroid = compute_centroid(S2_samples, c)

    S1_signature.append((S1_centroid, S1_count))
    S2_signature.append((S2_centroid, S2_count))

elif measure == FREQ:
  # Signature is a vector of tuples (index, abs(Fourier coefficient))
  S1_signature = []
  S2_signature = []
  S1_counts = []
  S2_counts = []
  for c in cuboids:
    # returns the number of samples from S1 in cuboid c
    S1_count = count_samples(S1_samples, c)
    # returns the number of samples from S2 in cuboid c
    S2_count = count_samples(S2_samples, c)
    if reduce_empty and S1_count == 0 and S2_count == 0:
      # Skip empty cuboid if reduce flag is set
      continue
    S1_counts.append(S1_count)
    S2_counts.append(S2_count)

  # Apply FFT and take absolute value
  # returns a vector of absolute values of Fourier coefficients for S1
  S1_coefficients = abs(FFT(S1_counts))
  # returns a vector of absolute values of Fourier coefficients for S2
  S2_coefficients = abs(FFT(S2_counts))

  # Generate and normalize indices
  n = length(cuboids)
  # returns a vector of n indices normalized to the range [0,1]
  indices = get_normalized_indices(n)

  # Create signature
  for i in range(n):
    S1_signature.append((indices[i], S1_coefficients[i]))
    S2_signature.append((indices[i], S2_coefficients[i]))

# Step 5: Compute dissimilarity by calculating the distance between signatures
    using the selected distance metric
dissimilarity = distance(S1_signature, S2_signature, metric=distance_metric)

```

```
78 return dissimilarity
```

Listing 3.5: The outline of the algorithm for dens and freq measures.

3.6.3 Parameter tuning

The values of the numerical parameters for the dens and freq measures were determined experimentally. Analogously to the shape measure, the aim of the parameter tuning was to find the threshold values of the parameters beyond which the dissimilarity value remains constant or nearly constant.

Fig. 3.8 illustrates the variation of the dissimilarity value for sample pairs of structures, calculated using the dens measure and various combinations of density and resolution parameters. The dissimilarity value tends to stabilize as the value of these two parameters increases. For most sample pairs, the stable value is reached when density=10 and resolution=8, therefore those parameter values were selected.

Fig. 3.9 shows how the dissimilarity value for sample pairs of structures varies with different combinations of density and resolution using the freq measure. This measure is more sensitive to the changes of the parameter values, resulting in larger dissimilarity variations than the dens measure. However, the overall pattern is that dissimilarity decreases as density and resolution increase. For consistency, the same parameter values as for the dens measure were selected: density=10 and resolution=8.

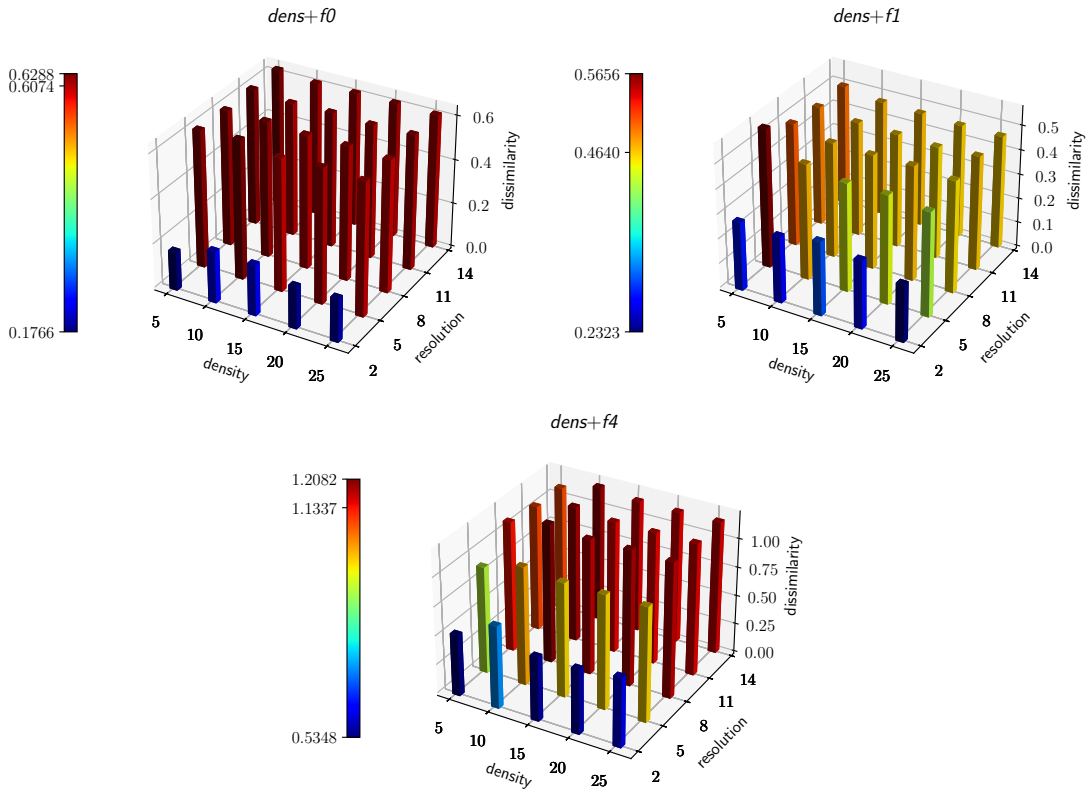


Figure 3.8: The effect of varying the density and resolution parameters on the value of the dens measure. The color of each bar represents the dissimilarity value.

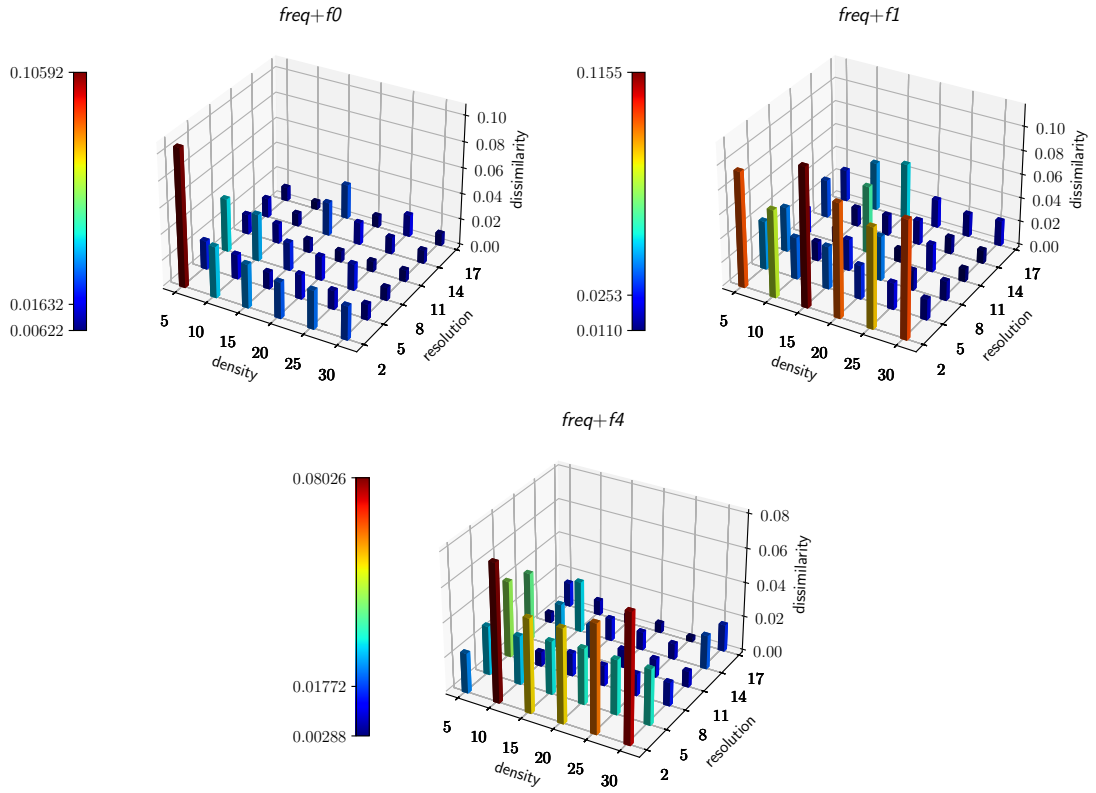


Figure 3.9: The effect of varying the `density` and `resolution` parameters on the value of the `freq` measure. The color of each bar represents the dissimilarity value.

3.7 Implementation

The `opt` and `shape` measures were implemented in `C++`, extending Framsticks SDK. A class named `SimilMeasureBase` was created to facilitate the implementation of subsequent dissimilarity measures. Classes that implement specific measures inherit from this class. Moreover, a parent class named `SimilMeasureMDSBased` was created for measures that need spatial alignment, such as `greedy` and `opt`. This class also inherits from the `SimilMeasureBase` class. The complete class hierarchy is shown in Fig. 3.10.

The `dens` and `freq` measures were implemented in Python as part of the `framspy` library. A single class named `DensityDistribution` is used for both measures. A boolean parameter `frequency` allows to select between the measures.

The source code of all implemented measures is publicly available at https://www.framsticks.com/svn/framsticks/.

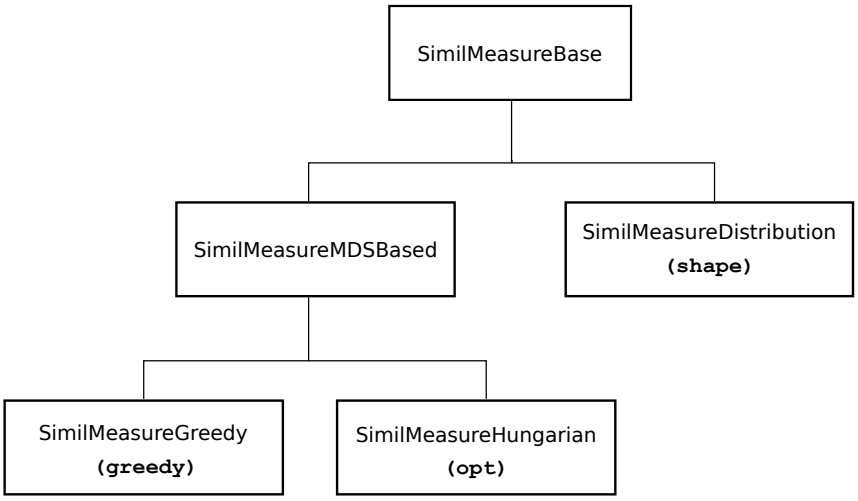


Figure 3.10: Class hierarchy for dissimilarity measures implemented in Framsticks SDK.

3.8 Dissimilarity measures comparison

Measure / Property	gene	greedy	opt	shape	dens	freq
Phenotypic	✗	✓	✓	✓	✓	✓
Generality	✓	✓	✓	✓	✓	✓
Extendability	✗	✗	✓	✗	✗	✗

Table 3.2: Comparison of the developed measures in terms of their domain (a genotype or a phenotype space), generality, and extendability.

Table 3.2 presents a summary of the characteristics of the proposed measures in terms of the domain they operate on (genotype or phenotype space) and the properties of generality and extendability. The `gene` measure is the only one that operates on genotypes and is used in this work, as stated earlier, to provide a reference for the phenetic measures. All considered measures exhibit the property of generality, as they can be applied to compare structures in any ED problem where solutions can be represented as undirected graphs. The `opt` measure is the only one that has the property of extendability, as it enables the incorporation of domain-specific information by including additional vertex properties into the distance calculation (the other measures would require modification of their algorithm to include additional properties). Another benefit of this measure is the ability to consider the distance between neural networks of active structures (although in a simplistic way). The measure also naturally provides high discriminability for structures represented as undirected graphs. However, it may perform poorly for solid-shapes models (Fig. 2.4), where each body part (shape) is treated by `opt` as a graph vertex and only its coordinates and vertex degree are considered and not its shape. For such models (not included in this work) `shape`, `dens`, and `freq`, which sample the structure's surface, might provide better discriminability.

3.8.1 Computational efficiency

To compare the computational efficiency of the measures, the three 400-element subsets of the `100_data_set` were used, each containing the structures evolved using different genetic encoding. For each such subset, a 400×400 dissimilarity matrix was computed for every dissimilarity measure. The experiments were conducted on a computer with an Intel Xeon Gold 5320 2.20GHz CPU and 256 GB of RAM.

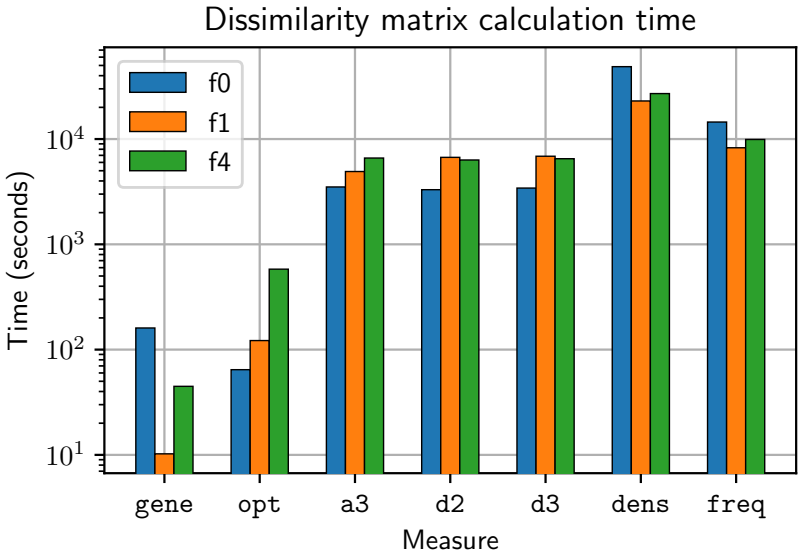


Figure 3.11: Calculation time of a 400×400 dissimilarity matrix for each genetic encoding and each dissimilarity measure.

Fig. 3.11 presents the results. The `gene` measure has the shortest computation times. For this measure, the *f0* and *f4* representations are the most computationally costly, due to their longer genotypes. The next fastest measure is `opt`. In this case, the *f4* and *f1* representations have the longest computation times, as the computation time depends heavily on the number of body parts (because of the Hungarian algorithm used in the matching procedure), which is on average higher for these representations. The measures `shape` (`a3`, `d2`, `d3`), `dens`, and `freq` are the most computationally intensive, as they involve creating a solid model of a structure, sampling it uniformly with points, and calculating Earth Mover's Distance (EMD). Additionally, unlike the `opt` and the `shape` measures which are implemented in C++, the two measures with the highest computation time, `dens` and `freq`, are implemented in Python.

3.8.2 Correlations between the dissimilarity measures

For the purpose of the measures comparison, the `100_data_set` (described in Sect. 2.4.1) was used. The subsets of the data set sharing the same genetic encoding were merged together, resulting in 3 sets of 400 genotypes each. These 3 data sets were used to compute Spearman's rank correlation coefficient between the dissimilarity values obtained from the considered measures, as well as to analyze their computational efficiency. Figs. D.1-D.3

in Appendix D show representative structures from each set. The shown structures were selected using the k-medoids algorithm [108] and the `opt` measure.

Initially, 10 dissimilarity measures were compared:

1. `gene`,
2. `greedy`,
3. `opt`,
4. `shape` with descriptor `a3`,
5. `shape` with descriptor `d1`,
6. `shape` with descriptor `d2`,
7. `shape` with descriptor `d3`,
8. `shape` with descriptor `d3`,
9. `dens`,
10. `freq`.

The parameter values of the measures were set as follows:

- `greedy` and `opt` measures:

  $w_V = 0$
  $w_D = 1$
  $w_N = 0.1$
  $w_G = 1$

- `shape` measure:

  `density`=10
  `bins number`=128
  `samples number`=10 000

- `dens` and `freq` measures:

  `density`=10
  `resolution`=8

The first step of the analysis of the relationships between the measures involved excluding some of them because of their high correlation with other measures in the set. Fig. 3.12 presents the scatter plots and correlation coefficients for the `shape` measure and the following pairs of descriptors: `d1`, `d2` and `d3`, `d4`. The Spearman's rank correlation coefficient for these pairs for all the encodings was in the range [0.99,1.0], showing strong correlation, therefore only one descriptor from each pair will be considered in further analyses: `d2` and `d3`. The `d2` descriptor was selected because the preliminary experiments on fitness-distance correlation analysis (FDC, see Chapter 4 for a detailed description of the FDC analysis) indicated that generally it provides better FDC values than `d1` (although the differences are minor). The `d3` descriptor was chosen because of its lower computational complexity in comparison to `d4`. Fig. 3.13 shows the scatter plots and correlation coefficients for `greedy` and `opt` measures. As expected, since the `opt` is an improved version of the `greedy`, the

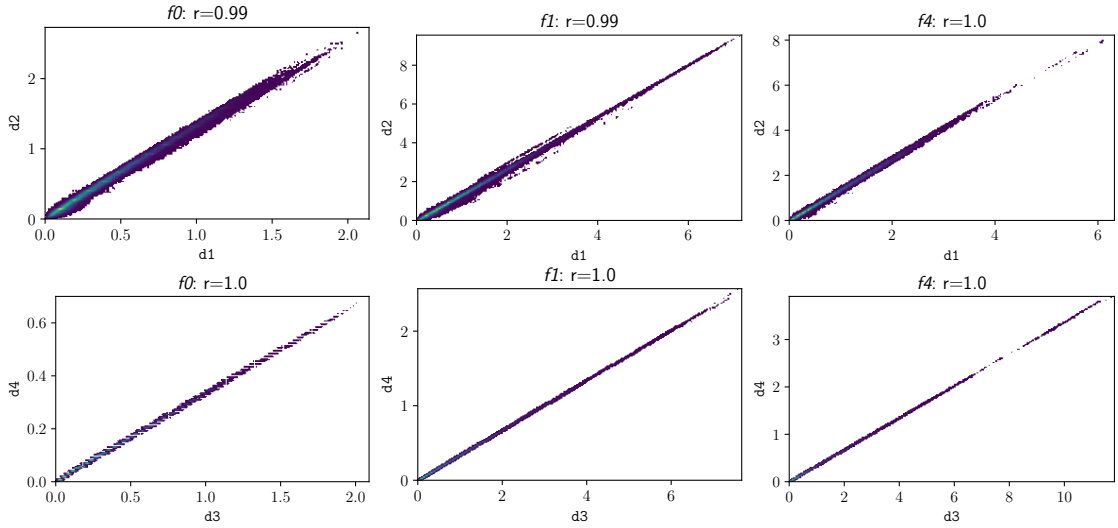


Figure 3.12: Strongly correlated descriptors of the shape measure: `d1` and `d2` (top row), `d3` and `d4` (bottom row). Each point on the plot represents the dissimilarity value for a pair of structures encoded using a given genetic encoding. The color indicates the density of points, with lighter colors corresponding to higher densities.

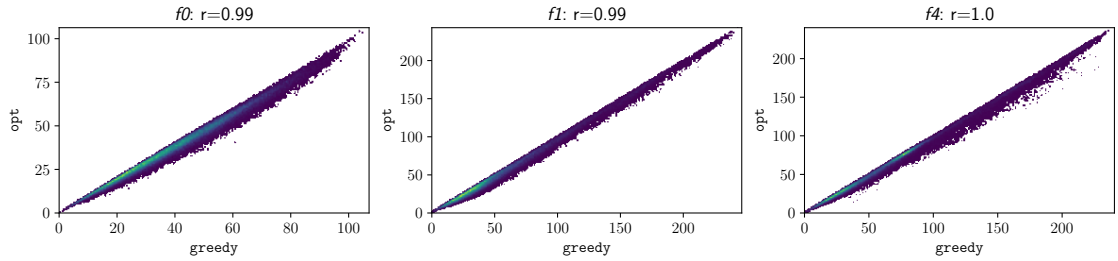


Figure 3.13: Strongly correlated measures: `greedy` and `opt`. Each point on the plot represents the dissimilarity value for a pair of structures encoded using a given genetic encoding. The color indicates the density of points, with lighter colors corresponding to higher densities.

dissimilarity values calculated using these two measures are strongly correlated. Therefore, in the following experiments, only the `opt` measure will be considered.

Figs. 3.14–3.16 show the Spearman's rank correlation coefficients for the dissimilarity values calculated using the considered measures, separately for each of the three subsets of structures. Figs. E.1–E.5 in the Appendix E show the scatter plots of dissimilarity values for each pair of measures. For the `gene` measure, the only genotype-based measure in the set, the correlations with other measures are in general weak. One notable exception is the correlation between `gene` and `opt` for the structures encoded using the *f0* encoding ($r = 0.73$). This may be attributed to the fact that `opt` is the only phenetic measure that also considers the neural network of structures, which is encoded in the genotype. Moreover, *f0* is a direct genetic encoding, which means that there is a higher correspondence between the genotype and the phenotype than in the other two representations.

The `opt` measure has generally moderate to strong correlations with other measures, except for the `freq` measure and the `shape` measure with `a3` descriptor, which correlates with `opt` weakly. Interestingly, the correlations are stronger for *f1* and *f4* genetic en-

Figure 3.14: Spearman's rank correlation coefficients for dissimilarity values calculated for structures encoded using *f0*.

codings than for *f0* (the strongest correlation is between the `opt` and `dens` for *f1* genetic representation: $r = 0.86$). These differences between encodings may indicate that the measures vary in how well they reflect the phenetic properties of the structures expressed by different genetic representations. It is also noteworthy that the correlation between dissimilarity values computed using different measures is not necessarily transitive: for *f0*, the correlation between `opt` and `gene` ($r = 0.73$) and between `opt` and `dens d3` ($r = 0.66$) is moderate to strong, but the correlation between `gene` and `dens d3` is weak ($r = 0.38$).

The measures that showed the strongest correlation across the representations were the measures based on the spatial distribution of structures, especially `shape` with `d2` descriptor and `dens` ($r \in [0.82, 0.9]$). This may imply that they are less fine-grained than the `opt` and `gene` measures.

3.8.3 Qualitative comparison

To further deepen the insights on the differences between measures, a qualitative comparison was performed. For this purpose, four structures from the `walking.gen` genotypes set (a part of the Framsticks distribution [126]) were used. The structures are depicted in Fig. 3.17. The dissimilarity matrices computed for the structures using `gene`, `opt`, `shape`, `dens`, and `freq` measures, are presented in Tables 3.3-3.7, respectively. According to all measures except for the `freq`, One-stick and Crawler are the most dissimilar structures. This congruence between the measures may be attributed to the fact that these structures

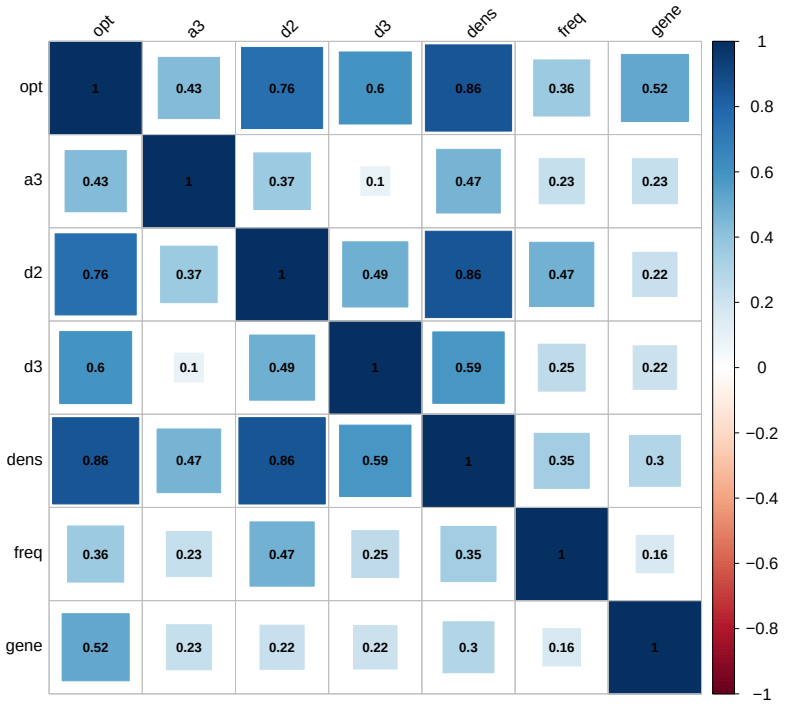


Figure 3.15: Spearman's rank correlation coefficients for dissimilarity values calculated for structures encoded using *f1*.

have the greatest difference in size (both in terms of a genotype and a phenotype). The only outlier, the `freq` measure, assigned the highest dissimilarity to One-stick and Spider. This, in turn, may reflect the lack of symmetry, the property present in the other three structures, in Spider.

The measures showed different results regarding the most similar structures, reflecting the different aspects of similarity that they capture. According to the genotype-based `gene` measure, One-stick and Quadruped are the most similar structures, since their genotypes require the fewest edits to transform one into another. The `opt` measure indicated that Spider and Quadruped are the most similar structures, since they have the smallest difference in terms of vertex degree, neurons, and geometric distance. Based on the overall shape, the `shape` measure indicated that Crawler and Quadruped are the most similar structures. Finally, the `dens` and `freq` measures identified Spider and Quadruped as the most similar structures, owing to their similarity in spatial distribution.

	quadruped	spider	one-stick	crawler
quadruped	0.00	116.00	101.00	107.00
spider	116.00	0.00	132.00	131.00
one-stick	101.00	132.00	0.00	133.00
crawler	107.00	131.00	133.00	0.00

Table 3.3: The dissimilarity matrix for sample structures computed using the gene measure. The minimum and maximum dissimilarity values are highlighted in green and red, respectively.

	quadruped	spider	one-stick	crawler
quadruped	0.00	15.66	32.65	22.64
spider	15.66	0.00	33.54	27.40
one-stick	32.65	33.54	0.00	48.92
crawler	22.64	27.40	48.92	0.00

Table 3.4: The dissimilarity matrix for sample structures computed using the `opt` measure. The minimum and maximum dissimilarity values are highlighted in green and red, respectively.

	quadruped	spider	one-stick	crawler
quadruped	0.00	0.41	1.07	0.30
spider	0.41	0.00	0.66	0.71
one-stick	1.07	0.66	0.00	1.37
crawler	0.30	0.71	1.37	0.00

Table 3.5: The dissimilarity matrix for sample structures computed using the shape measure using d2 descriptor. The minimum and maximum dissimilarity values are highlighted in green and red, respectively.

	quadruped	spider	one-stick	crawler
quadruped	0.00	0.49	0.82	0.64
spider	0.49	0.00	0.51	0.68
one-stick	0.82	0.51	0.00	1.06
crawler	0.64	0.68	1.06	0.00

Table 3.6: The dissimilarity matrix for sample structures computed using the `dens` measure. The minimum and maximum dissimilarity values are highlighted in green and red, respectively.

	quadruped	spider	one-stick	crawler
quadruped	0.00	0.01	0.04	0.02
spider	0.01	0.00	0.05	0.02
one-stick	0.04	0.05	0.00	0.04
crawler	0.02	0.02	0.04	0.00

Table 3.7: The dissimilarity matrix for sample structures computed using the `freq` measure. The minimum and maximum dissimilarity values are highlighted in green and red, respectively.

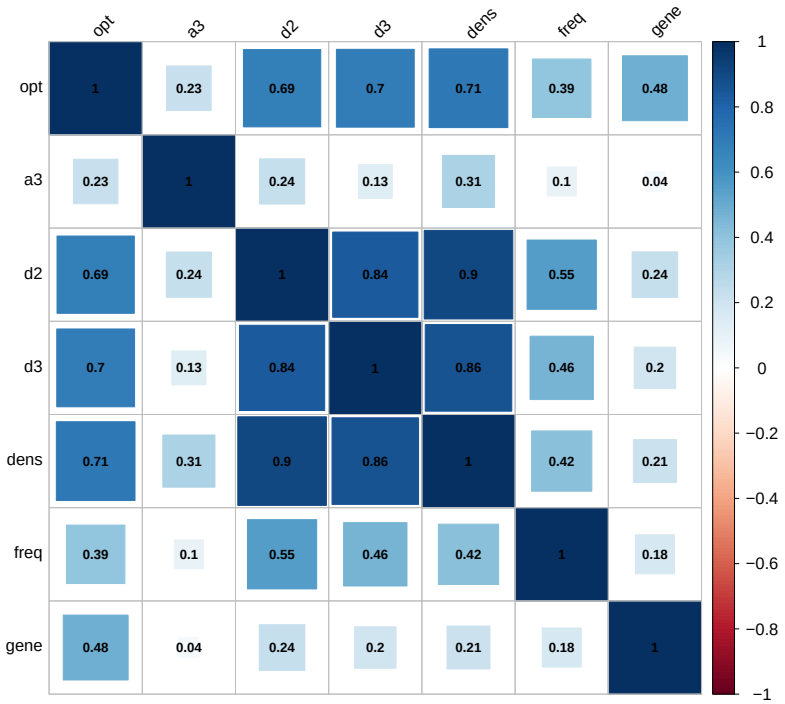
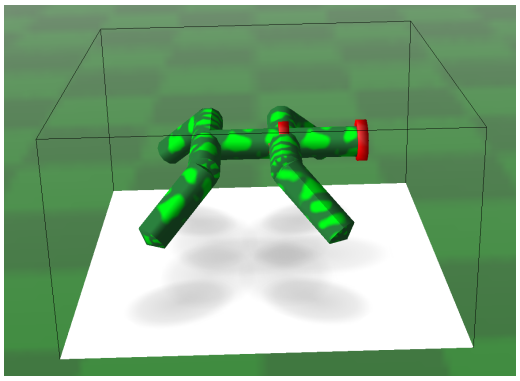
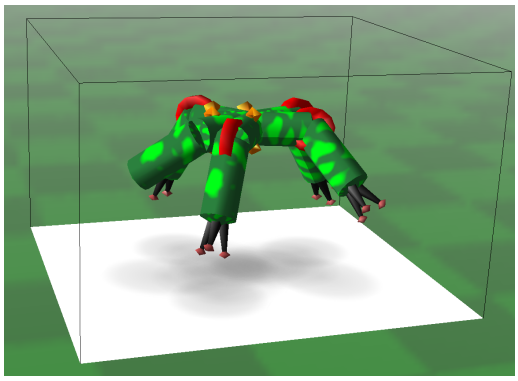


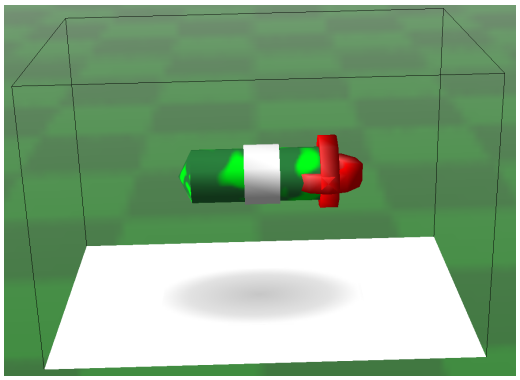
Figure 3.16: Spearman's rank correlation coefficients for dissimilarity values calculated for structures encoded using *f4*.



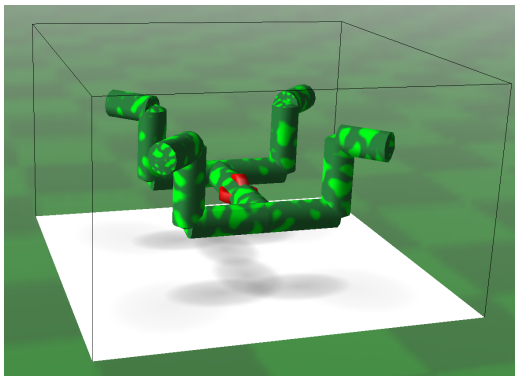
(a) Quadruped: MX[@*:−.4,2:−2.890,1:−1.808](RX(FX,),MX[@−1:1.849,0:3.094,0:−1.387][|−1:1.287,0:.5](rX(,X),,RX(X,)),rX(,FX))



(b) Spider: IIIX[S:4](,RccccXMMMX[|1:.5,T:−0.616],ccccRRXMMMX[|1:.1,T:5],,,RRccccXMMMX[|1:.1,T:5],IIcccX[S:4],RccccXMMMX,RRccccXMMMX[|0:.1,T:5])



(c) One-stick: X[|1:−0.782][@G:−741.994][G:294.200]



(d) Crawler: (X[Sin](rrX(rrrX(lX„)„),„rrX(RRRX(lX„)„))„X[N][−1:0.5,−2:0.265,si:0.5][|,−1:1,p:0.999][@,−4:0.366,p:1,p:1](rrX(rrrX(lX„)„),„rrX(RRRX(lX„)„)))

Figure 3.17: Structures used for the qualitative comparison of the dissimilarity measures.

3.9 Human perception of similarity

The study described in this section was previously published as a technical report [120].

3.9.1 Motivations

As discussed earlier in this chapter, one of the challenges posed by the development of dissimilarity measures for 3D structures is the lack of an objective ground truth to compare the dissimilarity assessments to. One of the solutions to this problem could be to minimize the Root Mean Square Distance (RMSD) [30] or a similar metric between two structures, but in most cases the computational complexity of this solution, arising from the infinite number of possible alignments of two structures, is prohibitive. Another approach is to use similarity assessments made by humans as ground truth. Employing humans in the evaluation of similarity has one key advantage: they can grasp features of 3D structures that make the structures similar due to human background knowledge and experience – i.e. humans can identify functional similarities just by looking at pairs of 3D objects.

The concept of similarity – as one of the fundamental elements of human perception and reasoning – has been widely studied, especially in the field of psychology [215, 4, 3]. However, the objects considered in these studies were vastly different from 3D structures modeled as graphs, considered in this work. In order to investigate the human perception of such structures and the factors that can potentially affect it, a study involving 14 participants was conducted, preceded by two pilot studies with a total of over 100 participants. To simplify the task for the participants, they were asked to evaluate the similarity rather than the dissimilarity between the structures.

3.9.2 Procedure

The investigation was performed using an online javascript application, which was internally using parts of Framsticks SDK [125] translated from C++ to javascript using emscripten [5]. Fig. 3.18 shows the main part of the user interface. The other part of the interface, located below the shown part, contained the personal questionnaire with drop-down lists for selecting answers.

In the first step, the instructor informed participants about the purpose of the investigation and the general experimental procedure. Next, the functionality of the interface was explained to them. Finally, the subjects were instructed to fill out the personal questionnaire, and then perform the following three steps, for each pair of the structures that would be displayed:

- slowly and carefully translate and rotate the two structures, until they are superimposed over each other in the best possible way (according to you); then, for each vertex of the smaller (in terms of the number of vertices) structure determine the most similar vertex from the larger structure,
- assess the similarity of the two structures on a scale from 0 to 100 percent,
- justify the way you matched the vertices in a concise manner.

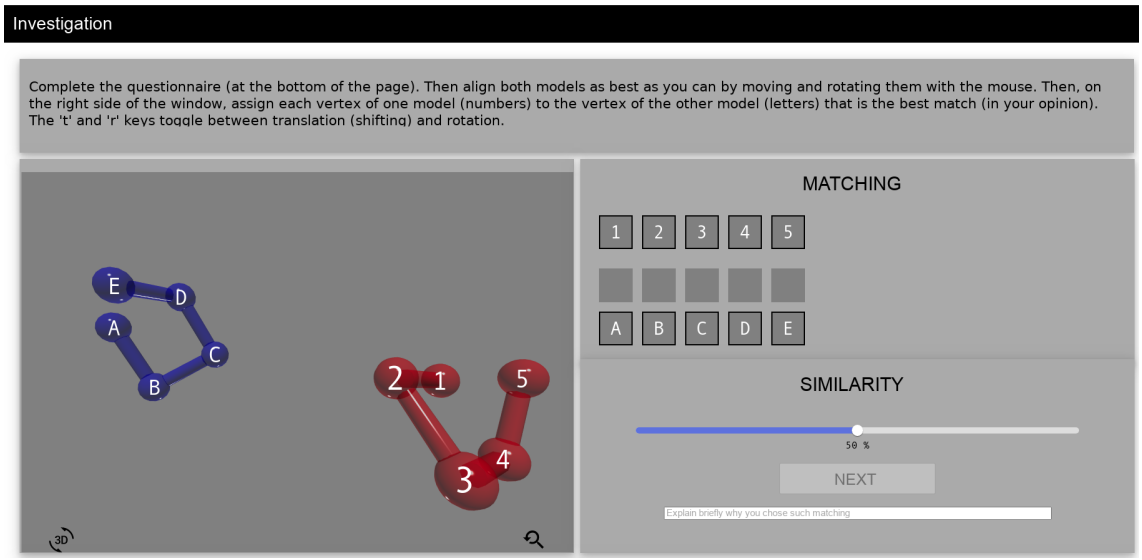


Figure 3.18: The main part of the user interface. The upper panel contains instructions. The left panel allows to translate and rotate each of the structures individually as well as the whole scene. Additional functionalities include zooming in and out, automatic centering of the scene and autorotation. The panel on the right is used for matching the vertices using the tiles, for similarity assessment using the slider and for inputting the text justifying the matching [120].

The following information (the factors that can possibly influence spatial cognition such as music education and learining foreign languages) was anonymously collected during the investigation:

1. Characteristics of the participants (for each participant):
   - gender [female, male],
   - age,
   - handedness [ambidexterity, left-handedness, right-handedness],
   - average sleep duration (in hours),
   - sleep duration on the night preceding the investigation (in hours),
   - number of years of playing video games,
   - average number of hours of playing video games per day,
   - level of musical education [amateur, none, professional],
   - number of years of music education,
   - number of foreign languages known,
   - number of years of sports training,
   - average number of hours devoted to sports training per week,
   - percentage of strength component in practiced sport,
   - percentage of endurance component in practiced sport,
   - percentage of an intellectual component in practiced sport,
   - conscientiousness (self-assessed on a Likert scale [104]),
   - proficiency in mental calculation (self-assessed on a Likert scale),

- proficiency in using maps and navigation (self-assessed on a Likert scale).

2. similarity assessment (for each pair of evaluated structures):
   - time of evaluation (in seconds),
   - position and rotation of each of the structures,
   - matching of vertices,
   - assessed similarity (percentage).

3.9.3 Pilot studies

The investigation was preceded by two pilot studies. The pilot studies differed from the final investigation in the following:

- number of participants,
- evaluated structures and their number,
- contents of the personal questionnaire,
- lack of verbal justification of vertices matching.

In both of the pilot studies the agreement between the subjects was measured using the following metrics:

1. Agreement in regard to the matching of vertices: the percentage of the vertices matched together in the same way for a given pair of structures by both subjects.
2. Disagreement in regard to similarity: the absolute difference in similarity ratings.

Pilot study I

In the first pilot study, 15 subjects assessed pairs of structures randomly selected from a genotype set consisting of 30 genotypes coming from the genotype set `walking.gen` distributed together with the Framsticks simulation environment [126]. The number of pairs that each participant had evaluated was not fixed, and the subjects were instructed to carefully assess as many structures as they could within a period of 10 minutes. The choice of the elapsed time as a condition for terminating the task rather than the number of evaluated structures was motivated by the expectation that in the opposite case, some of the subjects could perform the task quickly and inaccurately. In addition to matching and similarity assessment, the participants filled a short version of the personal questionnaire which concerned only their gender and the year of birth.

The subjects evaluated 59 pairs of structures in total. The mean number of assessed pairs per participant was 3.93, and its standard deviation was 3.28. The mean number of minutes devoted by a single subject to the assessment of a single pair of structures was 2 and its standard deviation was 1.93. From 59 evaluated pairs, only 9 were assessed by more than one participant, which allowed for only a limited analysis of the agreement between the subjects.

The participants differed significantly in regard to the matching of vertices. In 5 cases, the agreement was in the range $0-20\%$, in 1 case in the range $20-40\%$, and in 3 cases in the range $80-100\%$. Similarity assessments were significantly more consistent; 2/3 of the

absolute differences in similarity ratings were in the range $0 - 30\%$, and $1/3$ in the range $50 - 60\%$. In general, the subjects described the task itself as difficult.

Pilot study II

In order to get a better insight into the agreement between participants' assessments, in the second pilot study the number of subjects was increased to 73 and the size of the set of 3D structures was decreased to 8. As before, the structures came from `walking.gen`. Since in the previous pilot study the structures with a high number of vertices were especially hard for the participants to match, this time the structures were selected with the following distribution of the number of vertices: $2, 3, 4, 4, 5, 5, 6, 7$. Again, there was no fixed number of pairs to evaluate and the minimum duration of the task was set to 10 minutes, while the participants were encouraged to take as much time as they wanted to finish the task. The pairs of structures to evaluate were selected randomly. The personal questionnaire included all of the characteristics listed in Sect. 3.9.2, except for the percentages of different components in practiced sport, which were preceded by a simpler question regarding the type of practiced sport.

The subjects evaluated 318 pairs of structures in total. The mean number of assessed pairs per participant was 4.36 and its standard deviation was 2.5. The mean number of minutes devoted by a single subject to the assessment of a single pair of structures was 2.33 and its standard deviation was 2.28. In 288 cases a given pair of structures was assessed by more than one participant.

The results regarding the agreement in vertices matching and similarity assessment were similar to the previous pilot study. Fig. 3.19 shows the histograms of the percentage of identical matchings of vertices between subjects and the absolute differences in similarity ratings. Again, the participants were more consistent in evaluating the similarity of the

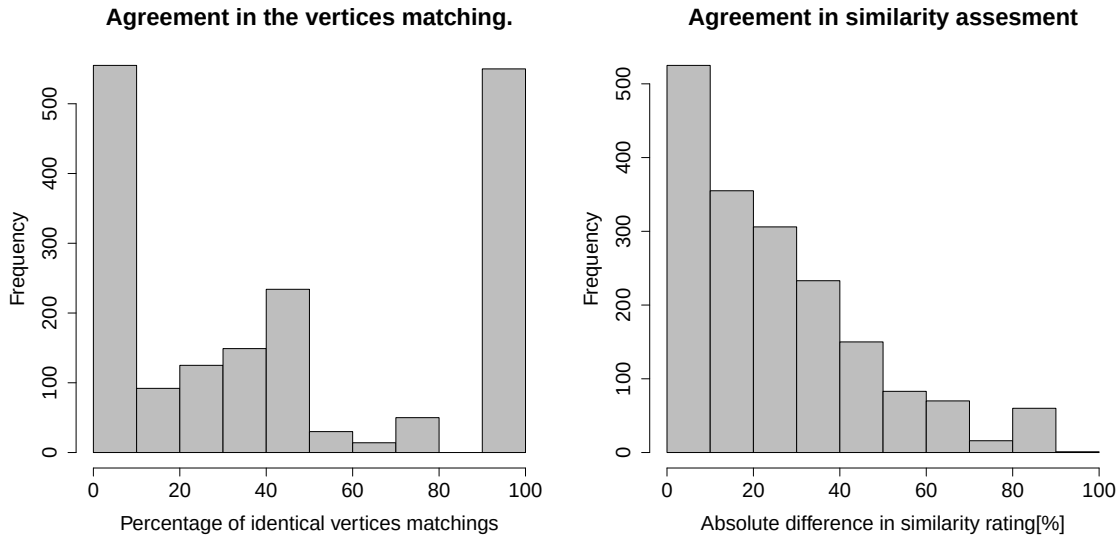


Figure 3.19: The percentage of the identical matchings of vertices between each pair of participants that assessed the same pair of structures (left column) and the absolute differences in similarity ratings between each pair of participants that assessed the same pair of structures (right column) in the pilot study [120].

structures than in matching their vertices. In order to assess the quality of the alignment of structures, for each of 318 evaluated pairs the Root-Mean-Square-Distance (RMSD) was computed using the following formula:

$$\mathrm{RMSD}(\mathbf{v},\mathbf{w}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}((v_{ix}-w_{ix})^2+(v_{iy}-w_{iy})^2+(v_{iz}-w_{iz})^2)}$$

where $\mathbf{v}$ and $\mathbf{w}$ denote the sets of the vertices from both structures that were matched together (unmatched vertices are not considered). Next, the normalized RMSD was calculated using the following formula:

$$\mathrm{RMSD}_{\mathrm{norm}}(\mathbf{s},\mathbf{t})_{\mathrm{j}} = \frac{\mathrm{RMSD}(\mathbf{s},\mathbf{t})_{\mathrm{j}}}{\mathrm{RMSD}_{\mathrm{min}}(\mathbf{s},\mathbf{t})}\cdot 100$$

where $\mathrm{RMSD}(\mathbf{s},\mathbf{t})_{\mathrm{j}}$ denotes RMSD calculated for the alignment of the structures $\mathbf{s}$, $\mathbf{t}$ made by j-th participant and $\mathrm{RMSD}_{\mathrm{min}}(\mathbf{s},\mathbf{t})$ denotes the minimal value of RMSD that was obtained for the structures $\mathbf{s}$, $\mathbf{t}$ across all participants; the minimal value of $\mathrm{RMSD}_{\mathrm{norm}}$ is equal to 100%. In order to compare the $\mathrm{RMSD}_{\mathrm{norm}}$ values across the participants, $mean(\mathrm{RMSD}_{\mathrm{norm}})$ was computed for all participants as the average of all $\mathrm{RMSD}_{\mathrm{norm}}$ values obtained by a single participant. The average $mean(\mathrm{RMSD}_{\mathrm{norm}})$ was equal to 4853.2% and its standard deviation to 11757.63%, showing significant differences between the subjects. Further analysis of the alignments made by the participants revealed that some subjects made little or no alignment at all.

3.9.4 Final study

The final study followed the procedure described in Sect. 3.9.2 and benefited from the experiences gathered in the two preceding pilot studies. In contrast to the pilot studies, each of the 14 participants assessed the same 5 pairs of structures to ensure a fully balanced data set. To prevent the subjects from omitting the alignment step, it was emphasized during the instructional part. Additionally, at the beginning of the investigation, the participants had to align two copies of the same structure; it allowed them to acquaint with the graphical user interface and get proficient in using its mechanics.

Again, the structures were selected or based on the genotypes from `walking.gen` in order to find out whether the matching of the vertices will be based on functional similarity, total distance minimization, or other strategies. To better understand the reasons behind the matchings made by the participants, their free-form textual justifications were collected as well.

In addition to the raw values of the time of evaluation of a single pair of structures and the RMSD, their normalized versions were considered just as in the pilot studies (see Sect. 3.9.3). Also, six simple features describing structures were created based on the two characteristics of the structures: the number of parts and dimensionality. The dimensionality was calculated as:

$$dim = \frac{bb_x}{max_{dim}} + \frac{bb_y}{max_{dim}} + \frac{bb_z}{max_{dim}} \tag{3.2}$$

where bb_x, bb_y, bb_z refer to the size of the structure's minimum bounding box in x, y

and z dimensions, and max_{dim} refers to the maximum of these values. The dimensionality dim takes values from nearly 1 (for structures that are significantly larger in one dimension than in the other two) to 3 (for structures that have equal sizes in all the three dimensions).

Evaluated structures

Fig. 3.20 shows the set of six structures from which five pairs were created; the pairs are enumerated in Table 3.8. The structures are in varying degrees creature-like – this was supposed to influence the matching of vertices so that in some cases the participants would base their choices on functional similarity, e.g. matching the "leg" of one structure to the "leg" of the other structure.

Participants

Figs. F.1-F.8 in the Appendix F show the distribution of the characteristics of subjects. 13 men and 1 woman aged from 23 to 28 years, all of them right-handed, participated in the investigation. The average sleep duration of the subjects was in the range from 7 to 9 hours and the duration of sleep on the night preceding the investigation was in the range from 0 to 9.5 hours, with most participants sleeping between 6 and 9.5 hours. The number of years of playing video games differed vastly – it ranged from 0 to 23 years, and the average number of hours devoted to playing video games per day ranged from 0 to 3 hours, except for one participant, who declared 6 hours.

The subjects were equally divided in terms of musical experience: 7 participants had no musical education at all and 7 participants were amateur musicians; their duration of musical education ranged from 1 to 10 years.

3 out of 14 participants did not practice any sports. For the rest of them, the number of years of practicing sports ranged from 1 to 10, except for one participant who declared 18 years; the average number of hours devoted to practicing sport per week ranged from 1 to 6, except for one participant who declared 10 hours. The type of sport declared by the subjects was usually a combination of endurance, strength and intellectual components, with the intellectual component being in most cases the smallest component.

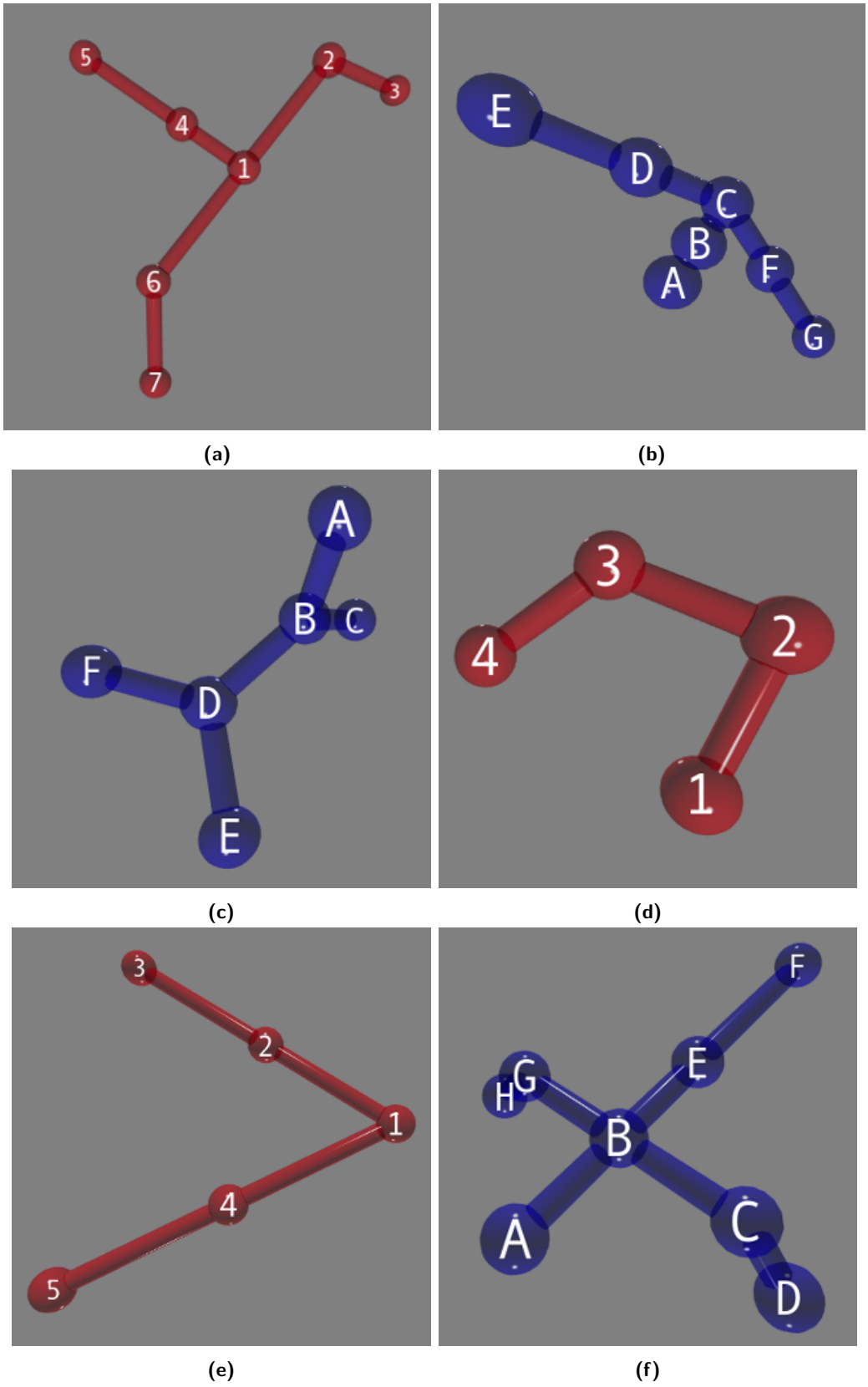
Most of the participants were undecided about whether they could describe themselves as a conscientious person. As to mental calculations and navigation skills, most of the subjects assessed themselves as rather proficient or proficient.

Time of evaluation

Fig 3.21 shows the distribution of the time of evaluation of a single structure. The most frequent evaluation duration was between 1 and 2 minutes, the cases in which it lasted more than 7 minutes were rare. From 5 evaluated pairs, the pair *af* had the lowest evaluation time, despite a relatively high number of vertices in both structures. This result can be

Pair number	1	2	3	4	5
Structure 1	a	c	a	b	a
Structure 2	b	d	e	f	f

Table 3.8: Pairs of the structures from Fig. 3.20 that were evaluated in the investigation.



(a) **(b)**

(c) **(d)**

(e) **(f)**

Figure 3.20: Six structures whose similarity was assessed in the investigation [120]. Each pair of structures was shown to participants with different labels for the vertices, numbers for one structure and letters for the other, to facilitate the matching task.

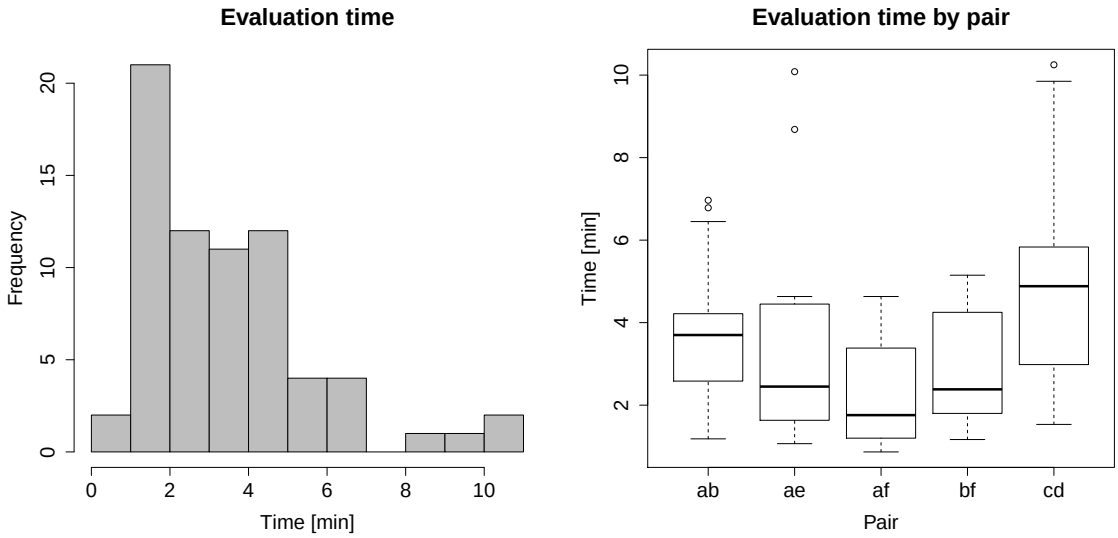


Figure 3.21: Distribution of the evaluation time of a single pair of structures, for all pairs of structures (left column) and the distribution of the evaluation time of a single pair of structures, depending on the evaluated pair (right column). The symbolic names of the structures constituting pairs are presented in Fig. 3.20 [120].

explained by the strong perceptual similarity between the structures and the relative ease with which they can be aligned. The evaluation lasted the longest for the pair *cd*, which could be caused by the atypical shape of the structure *c*, having its back and forward "limbs" in different planes and the lack of a way to closely align the two structures.

The time of evaluation was normalized using the same method as in the case of $\mathrm{RMSD}_{\mathrm{norm}}$ (see Sect. 3.9.3). No significant correlation between personal characteristics and the time of evaluation (raw or normalized) at the significance level of 0.05 was found. A significant negative correlation, at the significance level of 0.05, between raw evaluation time and minimum, maximum and average number of evaluated structures vertices was found, which can be explained by the ease of alignment of the structures pair *af*, which possessed the highest number of vertices.

Matching

Fig. 3.22 shows the distribution of the agreement between the participants in terms of the vertices matching. It can be seen that in general, as in the pilot studies, the subjects were not consistent between themselves; the median of the percentage of identical matchings was 50%. The agreement depended heavily on the specific pair that was evaluated: for the pair *af* it was equal to 100% for all the subjects, while for the pair *cd*, the median percentage of identical matchings of vertices was 0%. The reasons why those two pairs of structures were the easiest and the hardest to match are discussed in Sect. 3.9.4. A significant positive correlation between the agreement in the matching of vertices and the minimum, maximum and average number of vertex count was found, for the same reasons as for the time of evaluation, described above.

Table 3.9 compares matchings constructed by the `opt` measure with those constructed by humans. Since humans were not consistent between themselves in the matching of

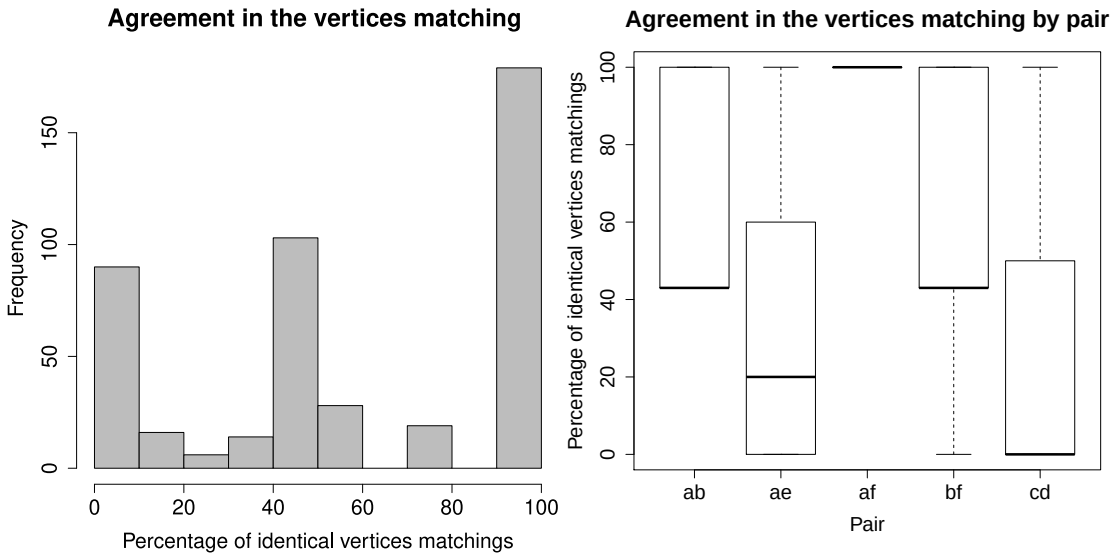
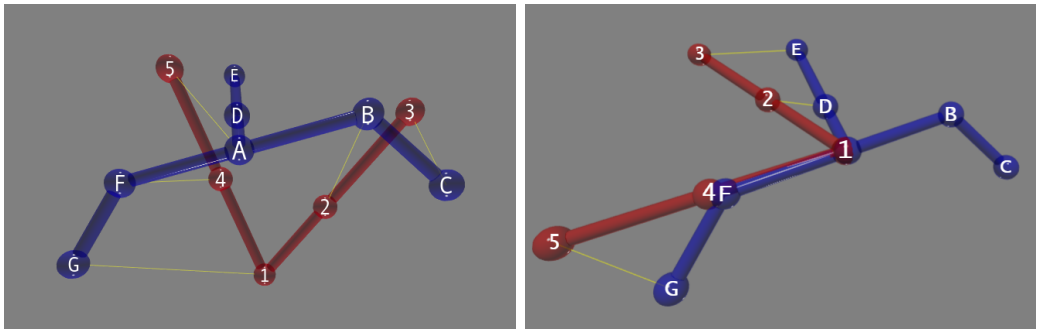


Figure 3.22: Percentage of the identical vertices matchings between each pair of participants that were assessing the same pair of structures (left column) and the same metric depending on the pair that was assessed (right column) [120].

pair	`opt` measure	humans	agreement [%]
ab	1:3 2:6 3:7 4:2 5:1 6:4 7:5	1:3 2:4 3:5 4:2 5:1 6:6 7:7 (57%)	43
cd	1:2 2:3 3:4 5:1	1:1 2:2 3:4 4:6 (29%)	25
ae	1:5 2:2 3:3 6:4 7:1	1:1 2:4 3:5 4:6 5:7	0
		1:1 2:2 3:3 4:4 5:5 (21%)	40
af	1:3 2:1 3:4 4:5 5:6 6:7 7:2	1:2 2:3 3:4 4:5 5:6 6:7 7:8 (100%)	57
bf	1:7 2:3 3:2 4:1 5:4 6:5 7:6	1:6 2:5 3:2 4:3 5:4 6:7 7:8	29
		1:6 2:5 3:2 4:7 5:8 6:3 7:4 (43%)	14

Table 3.9: Vertex matchings constructed by the `opt` measure and by humans. For humans, the most frequent matching for each pair was chosen; the percentage of subjects that made such a matching is shown in parentheses. In the case of ties, every matching that was made by a given number of participants is shown.



(a) `opt` measure **(b)** human

Figure 3.23: A sample matching of structures *a* and *e* constructed by the `opt` measure and by a human [120].

vertices, the most frequent matching made for each pair was chosen. It can be seen that human matchings differed significantly from the matchings made by the `opt` measure; the highest percentage of agreement (57%) was obtained for the *af* pair.

Fig. 3.23a shows the matching of vertices of the pair *ae*, and Fig. 3.23b shows a sample matching constructed for that pair by one of the participants. The first difference between the two matchings arises from the method of alignment used by the `opt` measure, which is based on the weighted Multidimensional Scaling (MDS) technique and which is applied to both structures separately, as described in Sect. 3.3.2. After applying MDS to the coordinates of the vertices, the center of a structure is located in the center of the coordinate system, the axis with the greatest variance becomes its first axis and the axis with the second highest variance becomes its second axis. In the case of human-constructed matching, the alignment is based rather on the similarity of the specific parts of the structures. The second difference is that the purpose of the `opt` measure is to minimize the total distance between the structures, including the penalty for the vertices that remained unmatched, hence the resulting matching may seem unintuitive for humans, who try first to match the most similar subgraphs.

Similarity

Fig. 3.24 shows the distribution of the similarity rating for each evaluated pair. The pair *af* turned out to be the most similar according to the participants, while the pair *ae* was assessed as the least similar. Fig. 3.25 shows the distribution of agreement between participants with regard to the similarity of structures. It can be seen that the participants were again more consistent between themselves than they were in the case of the matching of vertices; the third quartile of absolute difference in similarity rating was 33.5%. The agreement in similarity was not strongly dependent on the pair that was evaluated, as was the case with the matching of vertices. No significant correlation between personal characteristics listed in Sect. 3.9.2 (such as gender and age) and the agreement in similarity

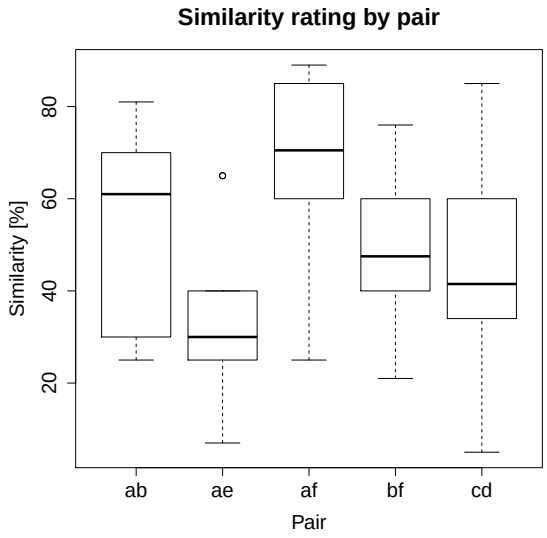


Figure 3.24: Distribution of the similarity rating for each of the evaluated pairs [120].

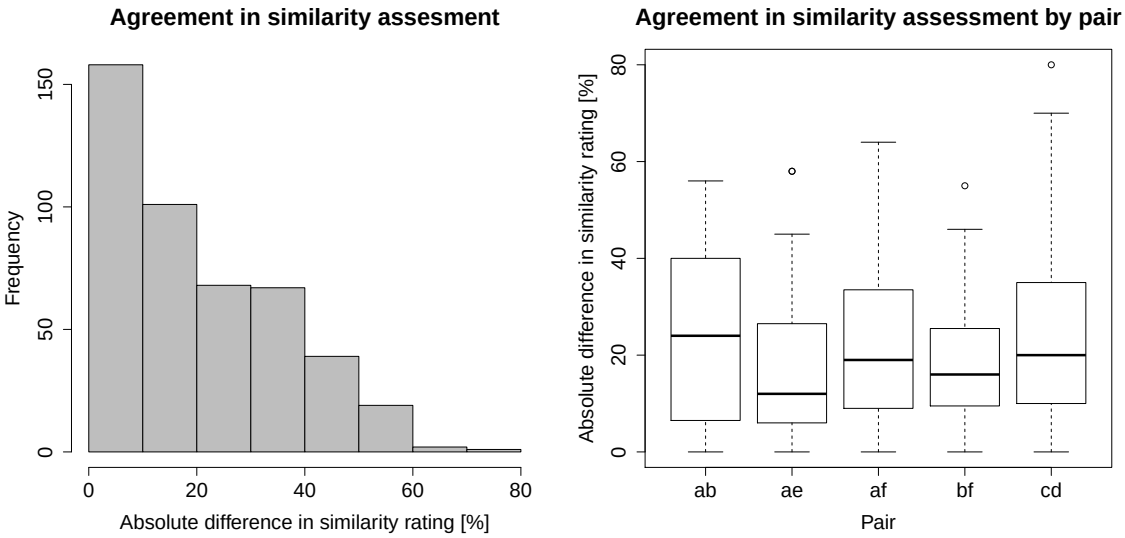


Figure 3.25: The absolute differences in similarity ratings between each pair of participants that were assessing the same pair of structures (left column) and the same metric depending on the pair that was assessed (right column) [120].

assessment at the significance level of 0.05 was found.

Table 3.10 shows the comparison of the similarity assessments made by humans and by the `opt` measure. The weights of the measure were set for the number of parts, vertices degree and geometrical distance to 1, and the weight for neurons to 0, since the neural networks were not presented to the human subjects. For a better comparison, the similarity ratings obtained from the `opt` measure were normalized to the range 0-100% and reversed by subtracting them from 100% (since the algorithm computes dissimilarity). Both the algorithm and the subjects indicated that the pair *ae* was the most dissimilar. The pair *ab* was the most similar according to the `opt` measure, followed by the pair *af*, while according to the human subjects that order was reversed. The pair *ab* obtained the highest similarity score from the algorithm, because the total distance between the matched vertices was the smallest, and there were no unmatched vertices.

pair	`opt` measure [%]	human [%]	human normalized [%]
ab	100.00	52.71	57.97
cd	60.95	46.07	38.15
ae	0.00	33.29	0.00
af	90.33	66.79	100.00
bf	38.32	48.93	46.69

Table 3.10: Similarity of each pair of structures evaluated by the `opt` measure and by humans. In the case of the `opt` measure, the similarities were normalized to the range $[0\%, 100\%]$ and reversed, in the case of raw human assessments, the average from all the participants is shown. In the last column human assessments normalized to the range $[0\%, 100\%]$ are shown.

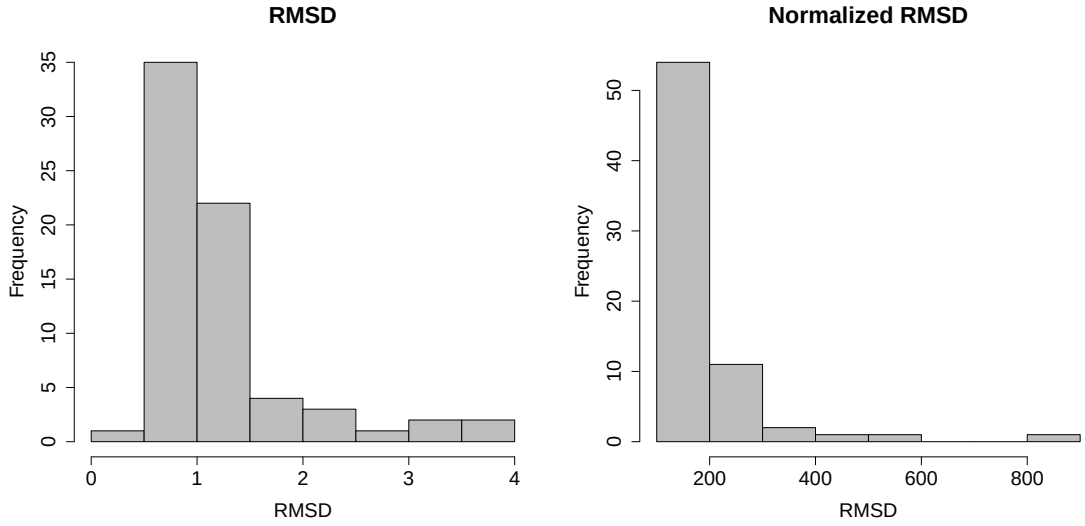


Figure 3.26: The distribution of raw RMSD (left column) and normalized RMSD ($\mathrm{RMSD}_{\mathrm{norm}}$) obtained for each pair of evaluated structures [120].

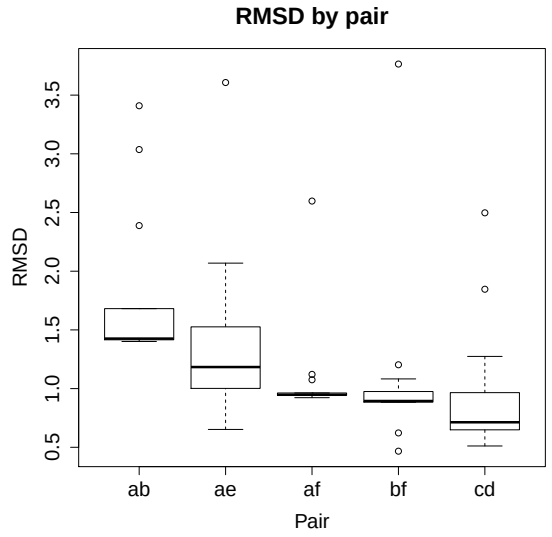


Figure 3.27: The distribution of RMSD depending on the pair that was evaluated [120].

RMSD

Fig. 3.26 shows the distribution of raw RMSD and $\text{RMSD}_{\text{norm}}$ between the matched structures. The differences in the RMSD values between participants were not as significant as in the pilot studies: 77% of the $\text{RMSD}_{\text{norm}}$ were below 200%, which means that most of the subjects did not obtain raw RMSD value which was more than twice as high as the minimum RMSD obtained for a given pair. No significant correlation between personal characteristics and $mean(\text{RMSD}_{\text{norm}})$ at the significance level of 0.05 was found. Another relationship that was investigated was the relationship between RMSD and dimensionality ((3.2)). A significant weak negative correlation between raw RMSD and the minimum, maximum and average dimensionality of structures was found, which again can be explained by the fact that the largest structures were the easiest to align.

Qualitative results

Table 3.11 presents the summary of the free-from descriptions that participants used to justify their matchings and similarity assessments. Most often the subjects tried to align the most similar parts of the structures together, and some of them were using expressions like “arms” and “tail” to refer to structural parts.

3.9.5 Discussion and conclusions

In this study, human subjects aligned five pairs of 3D structures, matching their vertices and assessing the similarity between the structures. Additionally, the personal characteristics of the subjects were collected. Responses of the participants of this study were compared to matching and similarity assessments performed by an algorithm.

Consistently throughout the two pilot studies and the main study, the participants were significantly more congruent between themselves in their similarity assessment than in the matching of vertices. Also, the agreement in the similarity rating did not depend so heavily on the specific pair that was evaluated as did the agreement in the matching

Vertex pair	Justifications
ab	The subjects indicated that the graphs representing the two structures have the same topology; they were trying to align the three “limbs” of the structures together.
cd	The subjects pointed out that there is no way to perfectly align the two structures; some of them were trying to perfectly align 3 out of 4 vertices of structure d at the expense of the fourth vertex.
ae	The structures were assessed as dissimilar, and the subjects were trying to align the most similar parts together.
af	The structures were assessed as similar, and the subjects declared that they aligned together corresponding “limbs”.
bf	The structures were assessed as partially similar, and again, the subjects were trying to align together corresponding “limbs”.

Table 3.11: Summary of the justifications of subjects for their matching of vertices and their similarity assessment.

of vertices. This result indicates that, to a large extent, the disagreement in the correspondence between certain structural elements does not influence the evaluation of their overall similarity. In general, the percentage of the same matchings was higher for the structures that were assessed as more similar with one notable exception: the pair *cd*, which was not rated as the most dissimilar, yet its matching of vertices caused the highest disagreement between participants. A possible reason could be the unconventional shape of the structure *c* that posed a challenge to the subjects during the alignment step.

In terms of the differences in RMSD values between the participants, there was a notable improvement compared to the pilot studies – most of the RMSD values obtained for a given pair of structures were lower than double of the minimum RMSD obtained for that pair. This improvement could be caused by emphasizing the alignment step during the experimental instruction stage, and by beginning the experimental procedure with the alignment of a pair of specifically prepared training structures.

Personal characteristics of the participants turned out to have no influence on the time of evaluation, the similarity rating, and the RMSD. However, it is worth noticing that the test group was quite homogeneous and consisted of only 14 people. No meaningful relationships between any aspect of the similarity assessment by humans (time of evaluation, RMSD calculated for aligned structures, the agreement in matching of vertices, and the agreement in the similarity assessment) and the features of the structures (the number of vertices and dimensionality) were found. It was likely caused by the limited set of the structures that were assessed, which was in turn forced by a significant cognitive cost of assessing the similarity of 3D structures and matching all their vertices.

The vertex matching constructed by the participants showed a significant discrepancy with the algorithmic matching, with an agreement ranging from 0% to 57% across different pairs of structures. However, the similarity assessments by humans and by the algorithm were much more consistent. Interestingly, while the algorithm rated the pair *ab* as the most similar, the human participants preferred the pair *af* that had a more functionally similar morphology.

3.9.6 Limitations of this study and future work

The study had some limitations that could be addressed in future work. One of them was the relatively small number of participants, which resulted from the high cognitive demand of the task of aligning 3D structures using a two-dimensional interface and interaction methods. Moreover, the participants were rather homogeneous, limiting the generalizability of the findings regarding the influence of personal factors. A potential improvement would be to use Virtual Reality technologies to facilitate the task and to recruit a more diverse sample of participants. Another limitation related to the high cognitive demand of the task was the small size of the set of 3D structures. To extract structural features that correlate with human similarity perception, a larger and more varied set of structures is needed.

Then, a similarity measure based on such features can be developed and applied in tasks such as designing genetic operators and measuring distance in diversity maintenance techniques. The results of the current study indicate that humans quantify similarity based on the perceived functional similarity. The structures that have high functional similarity

could potentially also have similar fitness in an optimization context. Therefore, using a similarity measure based on human judgments to define the neighborhood structure could enhance the search of the solution space. Further research and computational experiments are needed to verify this hypothesis.

3.10 Summary

In this chapter, six dissimilarity measures for 3D structures were introduced – one genetic and five phenetic ones. The genetic measure, `gene`, computes the Levenshtein distance between the genotypes. The first phenetic measure, `greedy`, is a graph-based heuristic measure that was natively implemented in Framsticks. The second phenetic measure, `opt`, is a measure developed as a part of this work that improves on `greedy` by finding the optimal vertex matching that minimizes the total distance and allows for incorporation of arbitrary vertex properties into the dissimilarity calculation. The third phenetic measure, `shape`, is another newly developed measure that adapts the shape descriptors and algorithm from Osada et al [166]. The last two phenetic measures, `dens` and `freq`, are also new measures, implemented as a part of this work, that compare the spatial distribution of the structures, either in the raw form or in the frequency domain.

The parameters of the measures, used in the experiments described in the following chapter, were determined by computational experiments. Then, the correlations between the dissimilarities obtained by different measures for the same structures were examined. This analysis led to the exclusion of the `greedy` measure, as it was strongly correlated with the `opt` measure. The results also suggested that different measures capture different aspects of dissimilarity between the structures. The qualitative comparison highlighted the features of the structures that each measure exploits, such as the overall shape for the `shape` measure and the presence of symmetry for the `freq` measure. The computational efficiency analysis demonstrated that the `gene` and `opt` measures are the most computationally efficient while the `dens` measure is the most computationally intensive. The investigation of human perception of similarity showed that humans tend to assess similarity of structures based on the perceived functional similarity, which is an encouraging result for the development of a dissimilarity measure based on human judgments, however further research is required.

The five selected measures: `gene`, `opt`, `shape`, `dens`, and `freq` will be used in the following parts of the work to the fitness landscape analysis, the development of the genetic operators, and in the diversity maintenance techniques for the evolutionary design of 3D structures.

4

Fitness-Distance Correlation Analysis

4.1 Introduction

The shape of a fitness landscape is one of the major factors determining the difficulty of an optimization problem. In general, smooth landscapes correspond to substantially easier optimization problems than rugged landscapes [198, 80]. Various measures have been proposed to quantify the ruggedness of the fitness landscape, such as correlation length, which is based on random walks in the solution space [201]. It is worth noting, however, that this measure is not invariant to the choice of genetic representation and operators, as they may affect the topology of the solution space and, consequently, the fitness landscape. Another useful concept for fitness landscape analysis is global convexity. In the landscapes exhibiting this property, better solutions are closer to each other and to the global optimum than worse solutions. This has been demonstrated for some combinatorial optimization problems [20]. Again, however, the fitness landscape may or may not possess the property of global convexity, depending on how the solution space is organized, for instance on the genetic operators used to traverse it [172] (see Fig. 4.1).

One way to assess global convexity is to analyze the correlation between the similarity of solutions and the similarity of their fitness values. A high correlation may indi-

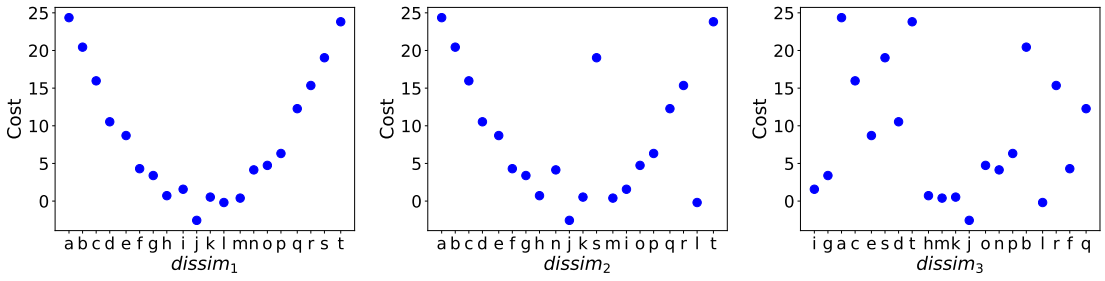


Figure 4.1: Three fitness landscapes for the same set of 20 solutions, represented by the letters 'a' to 't' [119]. Each of these fitness landscapes is a result of arranging the solutions according to one of the three dissimilarity measures: $dissim_1$, $dissim_2$, or $dissim_3$. The landscapes are ordered from the most globally convex (for $dissim_1$) to the most random (for $dissim_3$). The neighborhood and recombination operators should preserve the topology obtained for $dissim_1$ to improve the performance of the optimization algorithm and utilize the global convexity property during the search.

cate that the dissimilarity measure reflects fitness-related features of solutions. In that case, it should be feasible to design genetic operators that preserve these features and leverage the property of global convexity. This approach has been successful for combinatorial optimization problems, where distance-preserving crossover operators have been designed [98, 99, 132]. However, this approach has not received much attention in the context of optimizing active and passive 3D structures, due to, among other factors, the major challenge of developing dissimilarity measures for 3D structures. The objective of this chapter is to thoroughly analyze the global convexity of fitness landscapes for various ED tasks using five measures introduced in Chapter 3. It is a follow-up to the preliminary results on global convexity in ED tasks published in [119].

4.2 Experiments

4.2.1 Methods

To assess global convexity, the fitness-distance correlation (FDC) analysis can be used. FDC measures the correlation between the fitness of solutions and their distance from the global optimum (if it is known). It was introduced by Jones and Forrest as a method of investigating the optimization problems' difficulty [103]. Later it was also used to devise genetic operators in combinatorial optimization problems [98, 99, 132]. In this work, to investigate global convexity in the evolutionary design of 3D structures, the FDC analysis is conducted on the `1000_data_set` (detailed in Chapter 2). The set comprises 1000 structures for each of the following genetic encodings:

- *f0*,
- *f1*,
- *f4*,

and for each of the following optimization tasks (where the objective was subject to maximization):

- velocity on land,
- velocity in water,
- height of active structures,
- height of passive structures.

Five dissimilarity measures are used to calculate the distance:

- `gene`,
- `opt`,
- `shape` (descriptor `d2`),
- `dens`,
- `freq`.

The FDC is calculated as Spearman's rank correlation coefficient:

$$FDC = \frac{\sum_{i=1}^{n}(r_{fi} - \bar{r}_f)(r_{di} - \bar{r}_d)}{\sqrt{\sum_{i=1}^{n}(r_{fi} - \bar{r}_f)^2 \sum_{i=1}^{n}(r_{di} - \bar{r}_d)^2}},$$

where:

- n is the sample size,
- r_{fi} and r_{di} are the ranks of the fitness and distance values of the i-th solution,
- $\bar{r}_f$ and $\bar{r}_d$ are the mean ranks of fitness and distance.

The global optimum is unknown for all the optimization tasks considered in this chapter. Therefore, the distance to the global optimum cannot be used as a reference point for the fitness-distance correlation analysis. Instead, an alternative measure of distance is employed, which is the mean distance to solutions with equal or better fitness value [132]. This measure captures the proximity of each solution to the better solutions in the sample, and it can be computed for the i-th solution as follows:

$$d_i = \frac{1}{n}\sum_{j=1}^{n} dissim(x_i, x_j),\ x_j \in S_i$$

where:

- x_i is the i-th solution,
- S_i is the set of solutions that have equal or better fitness value than x_i,
- n is the size of S_i,
- $dissim(x_i, x_j)$ is the dissimilarity between solutions x_i and x_j.

A highly negative value of FDC in a maximization problem indicates that structures similar to the structures with equal or better fitness also exhibit high fitness, suggesting global convexity. Therefore, such a value is desirable.

The FDC was calculated for each combination of the genetic encoding and optimization goal, using each of the five measures. The following sections discuss the results for each of the employed measures.

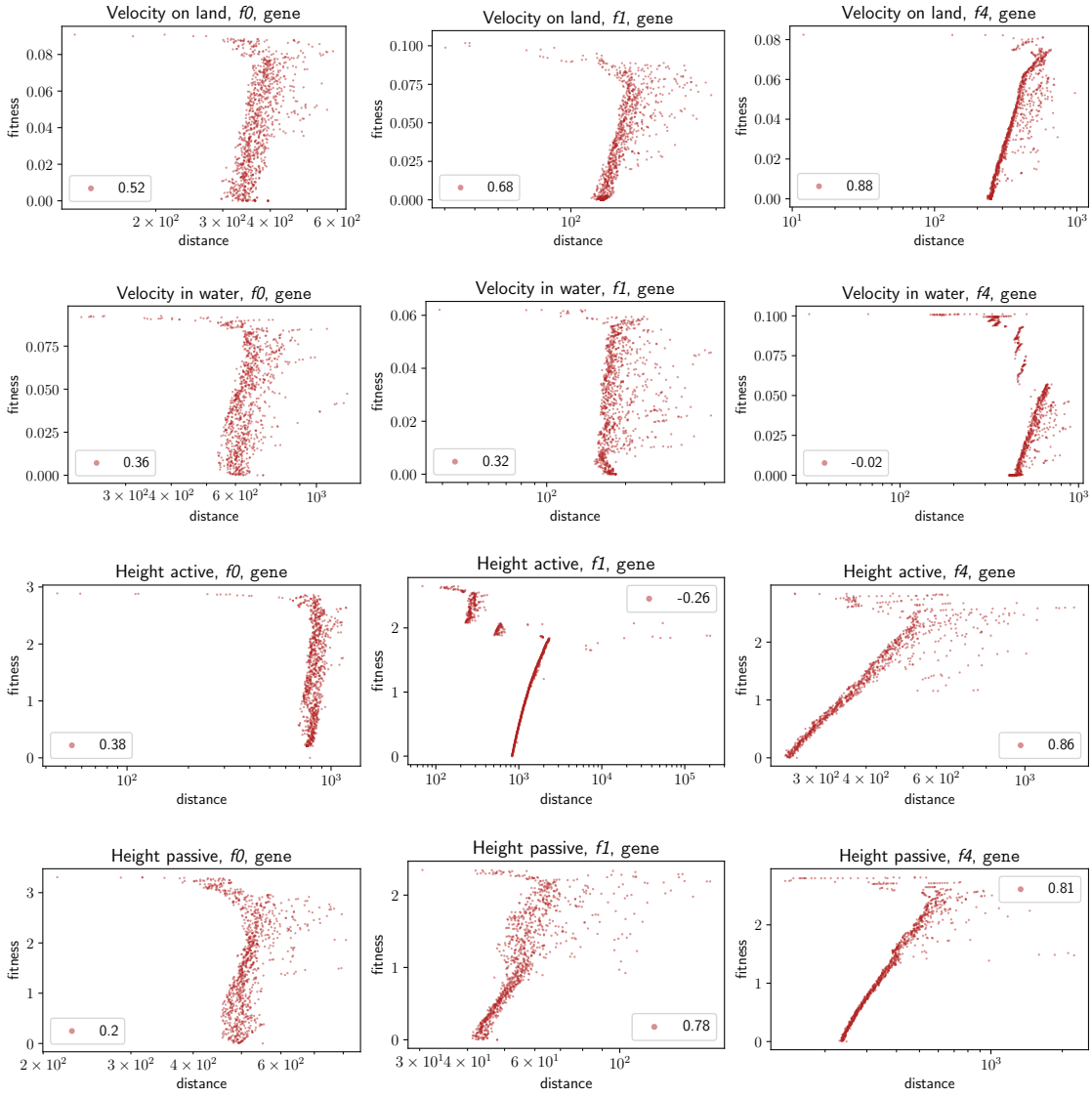
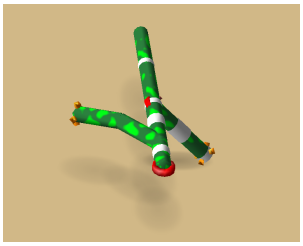


Figure 4.2: The relationship between fitness value of the structure and a mean distance (dissimilarity) to structures with the same or better (i.e. higher) fitness value obtained using the gene measure. Spearman's r_s rank correlation coefficient value is shown in the legend. Rows: optimization objectives, columns: genetic encodings.

4.2.2 The gene measure

/*4*/<QF<<X>qQN:G#3,>#2:−=::−/:>:−/:>ff<ff<fff<<LX#
2,#1>>M><rF<RmfQFLL<rmN:G>rlqFC,RCLqQf,MFqqN:Gpart
>,N:T>RmQLqQLL<<qX#2>>N:Gpart#3#1>>>LMXQ
><qXq,#3r,mCq,q,mLCMM#2>>QQ,#2FQcL,CF>qM
L><c<RMQQ<,cQcqmlLc<N:S>LFClq,fcQ„F,CmqcN:G:+=::−
=:#2,#3>>>RLffFLLMMq<RQLFCLCCqfN:|:−!::+!:[1:−2.30
8]:−!:#1:+/:#1:−=::−/::+/:>>>MFLN:G:+!:>rq,qLflM,<
qmN:S:−=::−/::−/:>N:Gpart>ffmfLL<rcLQcqqMLCFf,cN:G
:+!::+=:>RRCQMQLN:G>mCqqmN:N[0:−0.533][10:−2.436][
10:−1.991][9:2.598][2:9.5][10:2.611][1:−0.169][7:0.026]#3#1>
#1>>>MccN:N[5:−0.827][4:−0.896],[0:−0.761][4:−1.757][1:−4.91
5],:+/:#1>,>CCL<N:|[2:−0.138]><<qXm>,N:N[3:0.631][−1:−0.366]
[8:−0.606][3:−0.25]:+/:[3:6.772][0:−2.209][4:−1.351],#2#2>>>qC-
CQC<QfCX#2,mQ#2>C#2>>>RQlFQcqlN:S,#2>#2>>



/*4*/<QF<<<<X#2>>ffLcN:G#4,>#3:+!:>#1#3>>>f<fff<fff
<<LX#2,>>><MmfFQLL<<N:G>N:G>qqFqCqN:Gpart
#2>>RQLqQLLM<<qX>N:Gpart#2#3#5>>>>L#4#2>
>MX,#3M#2#2m>m>>,><X#3rmMqqLc,>Q#2MFQcLC
QF#1„#4,>>#2>>qMQ,L><R<Mc<Ccq<LC,RFCLFM
N:G>N:S>rLLMMq<LCFF,rFfN:|:−!:[1:−2.308]:−!:#5:+/:
:−=::+/:#3>>,#1,#2#3:−/::+/:>,,>>>N:G>rlfQCqN:S>qq
q<CQLN:G>lN:G,#3,>>CN:N[2:−0.496][4:1.048],[0:6.689]#2[9:−1.
962][10:0.143][10:0.197]>>CN:N[4:0.906][−4:4.357][0:0.123][−2:−3.936][
3:0.817][5:−1.605]:−!:[7:1.728][3:5.88]>MN:Gpart#1:−!::+=::+
!:,#2,#2>>„#5>>#2,:+!:>>MN:Gpart:+/::−=:#3„#4>,#
4>#2,>:+!:>>CCL<N:|[2:−1.5]:−=:><<qX>N:N[15:−1.551][3:3.449]
[9:−0.244][11:−2.538],[13:5.798]:−!:[7:−3.997][7:2.718]:−=:#2[−1:−6.94
3]>>QCQM<QfMmXm>MFQN:S>

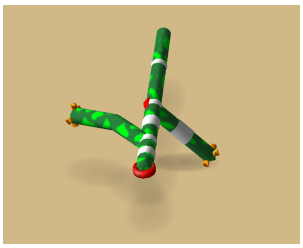


Figure 4.3: Two structures encoded using the *f4* encoding, evolved in the *velocity on land* task. The structures have the same morphology (right column) and very similar fitness (approximately 0.073), but different genotypes (left column). As a result, the dissimilarity between these structures, computed using the gene measure, is high (495).

Fig. 4.2 illustrates the correlation between fitness values and mean distance to structures with the same or better fitness, measured by the gene measure. In most cases, except for the *f1* encoding in the *height passive* task and the *f4* encoding in the *velocity in water* task, the correlation coefficients between fitness and distance are positive, ranging from weak to strong. The scatter plot shown in Fig. 4.2 indicates that for each task and each genetic encoding, a small proportion of solutions exhibits low distance and high fitness, but the majority of the solutions follow the opposite trend (i.e. the higher the fitness the higher the distance). This means that the high-fitness structures tend to be highly dissimilar from the structures with the same or better fitness in terms of the gene measure. Several factors contribute to this result. First, all considered genetic encodings are redundant, so different genotypes can encode the same phenotype, thus structures with the same morphology and fitness may vary in their genotypes. Second, the gene measure treats digits present in genotypes as characters, so it may not capture the similarities between morphologies expressed using *f0* and between neural networks in all three encod-

ings because the information about numerical distance is disregarded. Third, the more fit solutions generally tend to have more body parts and neurons, which results in longer genotypes. For the solutions with similar morphologies and long genotypes, the number of edits to transform one genotype into another may be high and hence the resulting distance may also be high. Fig. 4.3 shows an example of such structures. On the other hand, small, low-fitness solutions will be on average more similar to the solutions with the same or better fitness, as the number of edits required to transform one genotype into another is smaller.

4.2.3 The opt measure

Fig. 4.4 illustrates the correlation between fitness values and mean distance to structures with the same or better fitness, measured by the `opt` measure, broken down by $\mathtt{d_D}$, $\mathtt{d_N}$, and $\mathtt{d_G}$ components (see Sect. 3.3). This measure generally outperforms the `gene` measure in terms of FDC. However, the results vary significantly depending on the genetic representation and optimization goal. For example, using the *f0* representation in the *height passive* task, there is a strong monotonic relationship between fitness and distance for the $\mathtt{d_D}$ component, resulting in $r_s = -0.86$. A similar negative correlation is observed for the $\mathtt{d_D}$ component using the *f0* encoding in the *height active* task. In contrast, the FDC is positive for these tasks using the other two genetic encodings and near-zero for the *f0* encoding in other optimization tasks. For tasks in which the velocity of structures is maximized, the strongest negative correlations are observed for the *velocity in water* using the *f1* representation, again for the $\mathtt{d_D}$ and $\mathtt{d_G}$ components. In other cases, the distance in terms of the difference in vertex degree and geometry does not correlate with fitness. However, there is a moderate negative correlation between the difference in neuron number and fitness for the *velocity in water* task using all three representations and the *velocity on land* task using the *f1* representation.

For the *height passive* task using *f1* and *f4* representations, FDC values computed for $\mathtt{d_D}$ are weakly positive. As shown in Fig. 4.4, there exist clusters of highly-fit solutions that differ in distance to the same or better solutions. Fig. 4.5 shows two sample structures from different clusters that are similar in terms of fitness but highly dissimilar in terms of distance calculated using $\mathtt{d_D}$ component. Therefore, unlike in the case of the *f0* representation, the correlation between distance and fitness does not indicate global convexity for the *f1* and *f4* encodings, as there are multiple regions of high fitness that are not close to each other in terms of the `opt` measure.

4.2.4 The shape measure

Fig. 4.6 illustrates the correlation between fitness values and mean distance to structures with the same or better fitness, measured by the `shape` measure using `a3`, `d2`, and `d3` descriptors. Overall, the `d2` descriptor yields the best results, with the most negative FDC values. This means that structures with higher fitness are closer to each other according to this descriptor.

The highest negative values were achieved in tasks in which the vertical position of structures is maximized. For the `d2` descriptor the fitness generally decreases with the

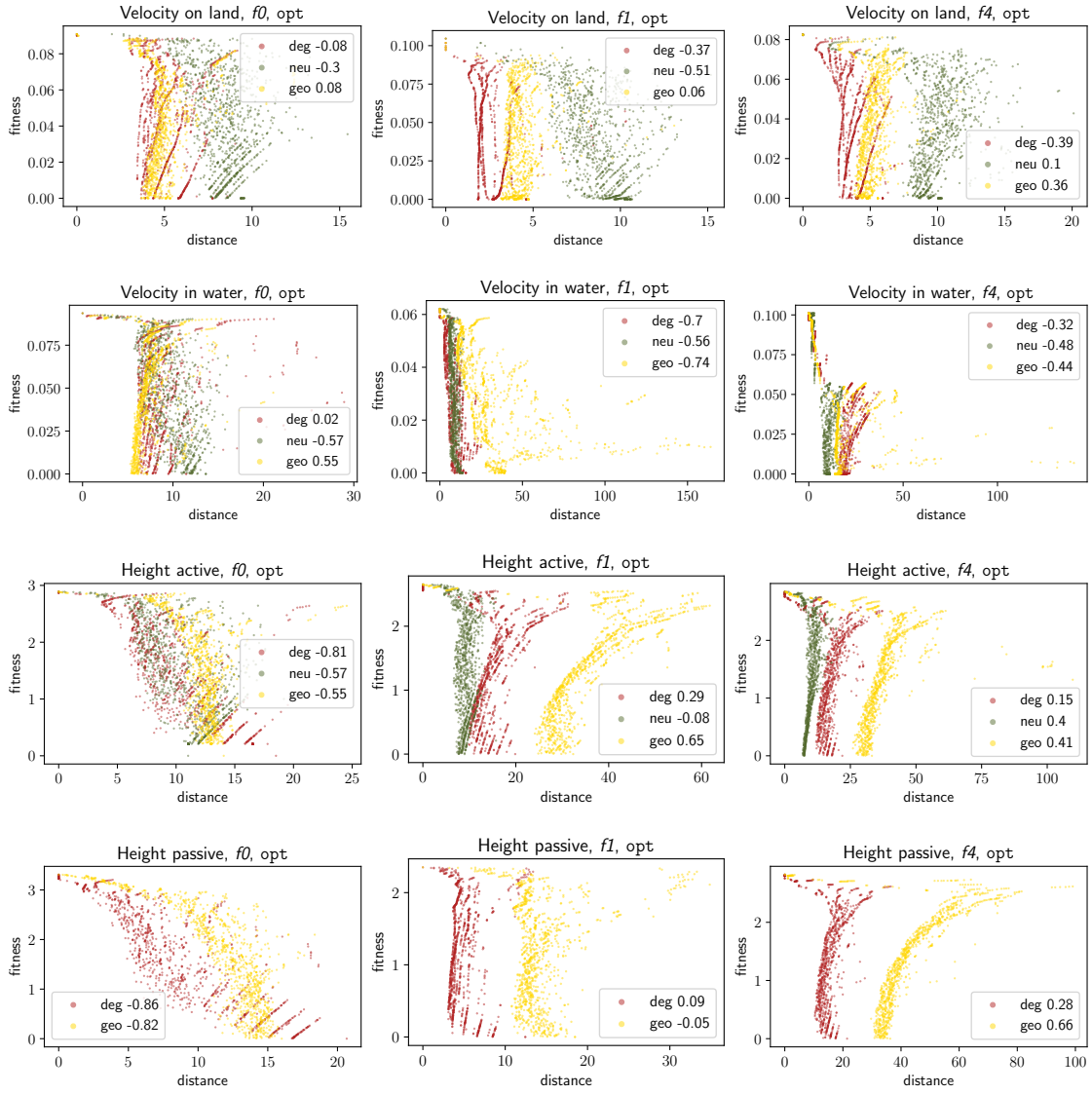
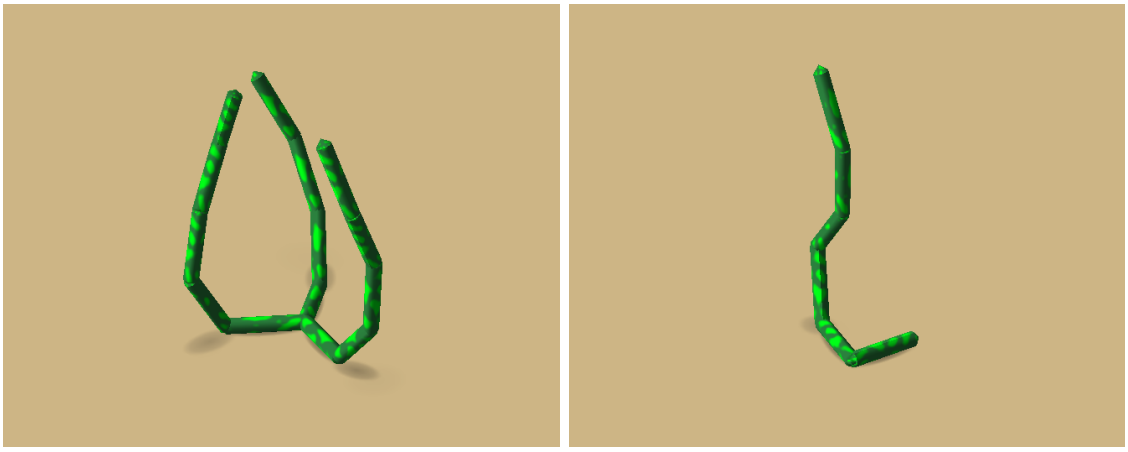


Figure 4.4: The relationship between fitness value of the structure and a mean distance (dissimilarity) to structures with the same or better (i.e. higher) fitness value obtained using the opt measure. Results for each component of the dissimilarity measure are presented separately on each plot, except for the d_V component, which is not taken into account. Spearman's r_s rank correlation coefficient values are shown in the legend for each of the components. Rows: optimization objectives, columns: genetic encodings.

increase of the distance. The FDC values range from -0.66 to -0.9. This may reflect the fact that in those tasks fitness is strongly related to the shape of the structure. The highly negative FDC values were also observed for the d2 descriptor in the *velocity in water* task using the *f1* encoding, where longer structures tend to have higher fitness. In other tasks that involve velocity, the correlations are weaker, which can be attributed to the fact that fitness can depend equally or more on the neural network and the strategy of locomotion.



(a) Vertical position: 2.09 **(b)** Vertical position: 2.08

Figure 4.5: Two passive structures with similar fitness, yet high distance in terms of difference in the vertex degrees ($d_D = 16$).

4.2.5 The `dens` and `freq` measures

Fig. 4.7 illustrates the correlation between fitness values and mean distance to structures with the same or better fitness, measured by distribution-based measures `dens` and `freq`. The `freq` measure, which uses the frequency domain representation of the structures' spatial distribution, yields better FDC values than the `dens` measure, which uses the raw distribution, in all tasks except the *velocity in water* task using the *f1* encoding. The FDC values for the `freq` measure are better (more negative) in tasks that involve velocity, ranging from -0.43 to -0.75, than in tasks that involve vertical position, ranging from -0.45 to -0.78. The `dens` measure only yields a highly negative FDC value for the *velocity in water* task using the *f1* encoding, similarly to the `shape` measure using `d2` descriptor.

These results may imply that the `freq` measure captures the properties of structure that correlate with fitness better than the `dens` measure. It is noteworthy that the `freq` measure performs better than the `shape` measure, discussed in the previous section, in tasks that involve velocity, except for the *velocity in water* task using the *f1* encoding. However, for the task in which vertical position was maximized, the `freq` measure provided worse FDC values than the `shape` measure.

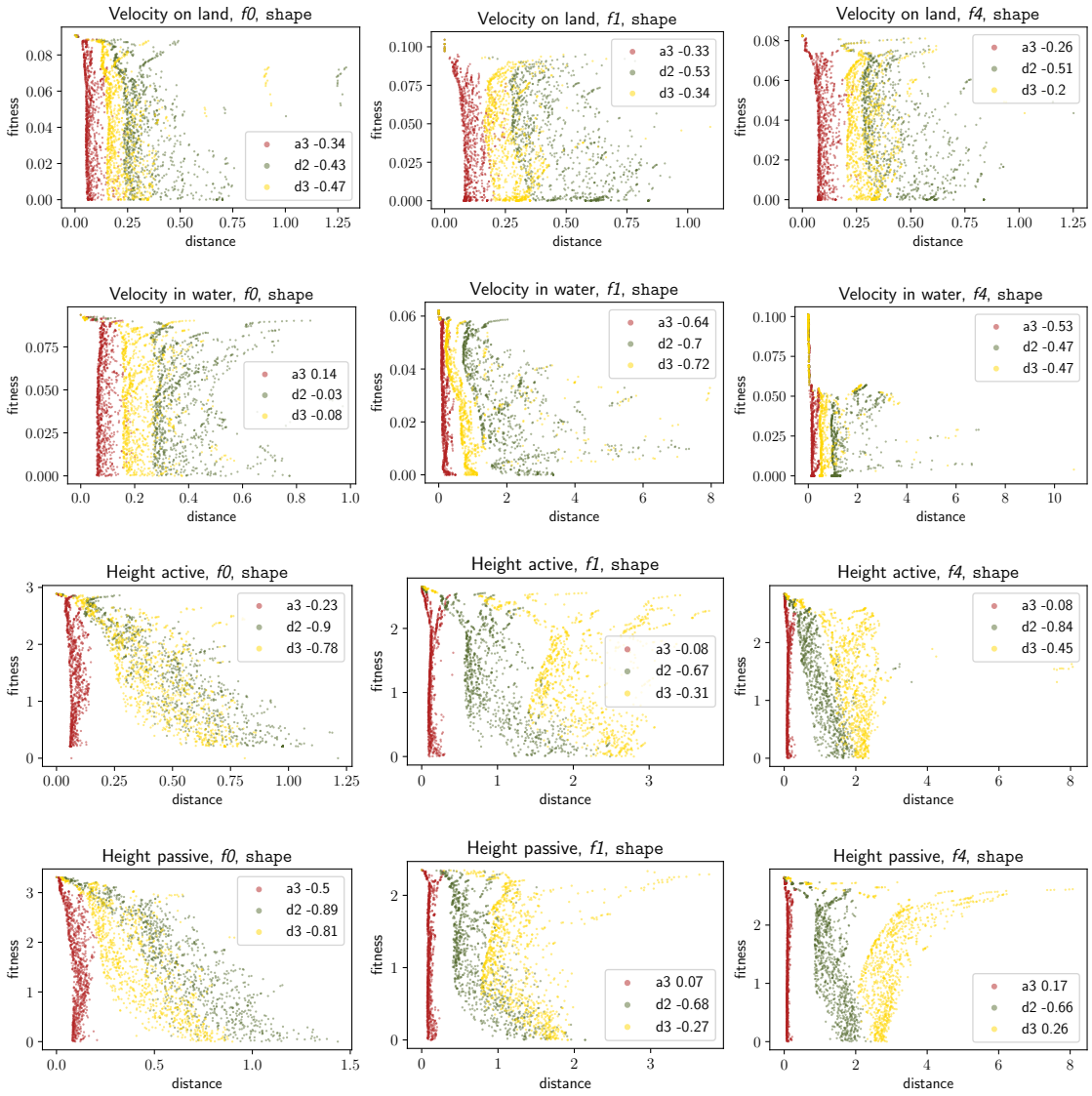


Figure 4.6: The relationship between fitness value of the structure and a mean distance (dissimilarity) to structures with the same or better (i.e. higher) fitness value obtained using the shape measure. Results for each descriptor taken into account are presented separately on each plot. Spearman's r_s rank correlation coefficient values are shown in the legend for each of the descriptors. Rows: optimization objectives, columns: genetic encodings.

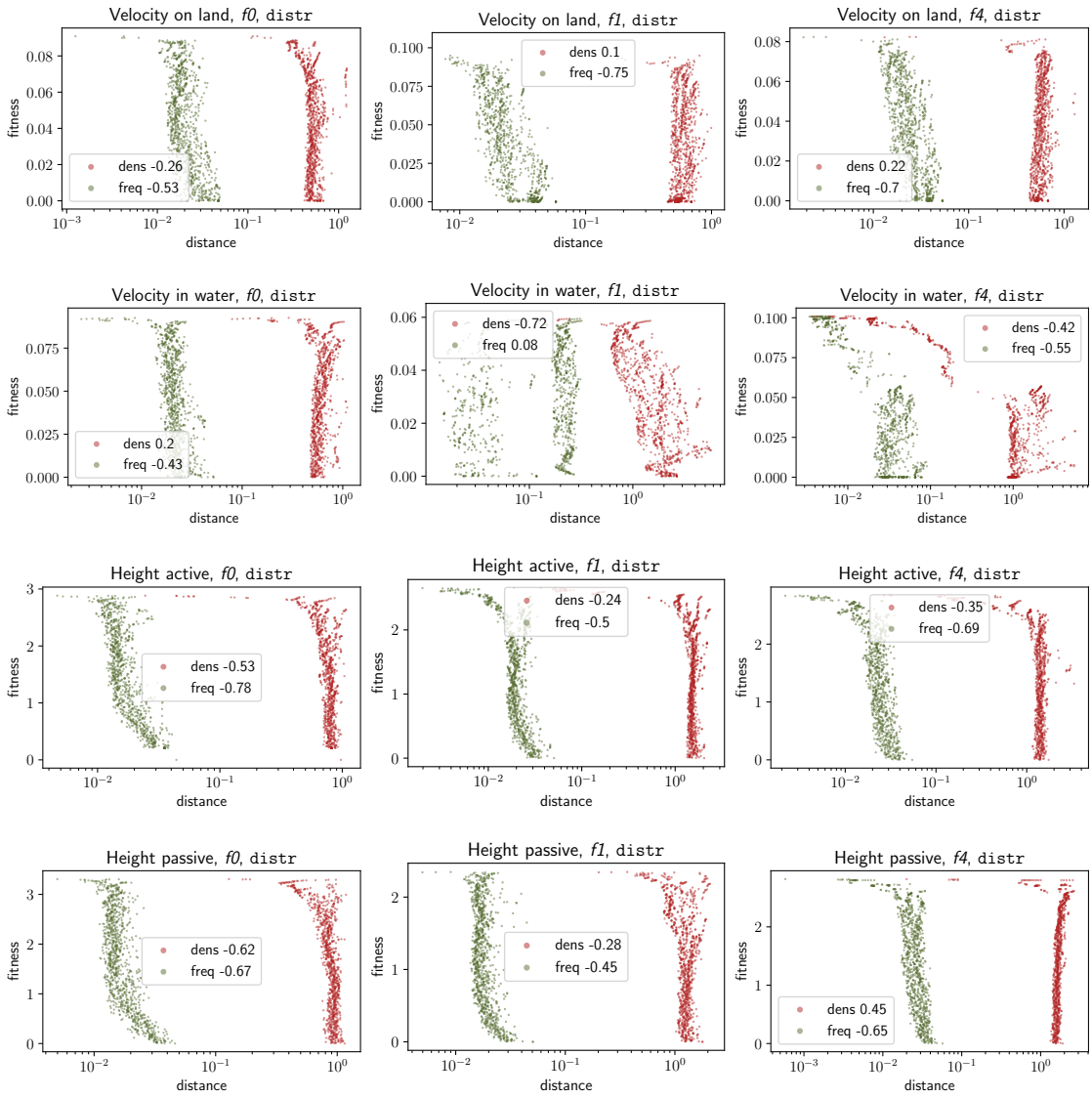


Figure 4.7: The relationship between fitness value of the structure and a mean distance (dissimilarity) to structures with the same or better (i.e. higher) fitness value obtained using dens and `freq` measures. Spearman's r_s rank correlation coefficient values are shown in the legend for each of the measures. Rows: optimization objectives, columns: genetic encodings.

4.3 Summary

This chapter examined the global convexity of fitness landscapes in four optimization tasks using three genetic representations. The global convexity was assessed by the fitness-distance correlation (FDC) analysis using five dissimilarity measures: one genetic and four phenetic. The results revealed that FDC values varied depending on the genetic representation, the optimization task, and the employed dissimilarity measure. The genotype-based measure `gene` yielded mostly positive values that do not indicate global convexity of the fitness landscape. For the phenetic measures, in general, better (i.e. highly negative) FDC values were observed in the task involving maximizing the height of the structures, in which the influence of the morphology on the fitness value is stronger than in the velocity-related task. However, even within the same task and genetic representation, the results differed depending on the employed dissimilarity measure. For instance, for the *height active* task using the *f4* representation, the values yielded by the `opt` measure were positive while those yielded by the `shape` measure using `d2` descriptor were highly negative.

The overall pattern was that better (highly negative) values of FDC were observed for the shape- and distribution-based measures, `shape` and `freq`, especially for the *height active* and *height passive* tasks. The FDC computed using the morphology graph-based measure `opt` was in some of those tasks positive and indicated a multi-modal fitness landscape. These results may suggest that `shape` and `freq` measures capture the properties of the structures that correlate with fitness and thus are better suited to be used in genetic operators that aim at preserving the traits of parents in offspring solutions. On the other hand, it may also imply that `shape` and `freq` are less fine-grained than the `opt` measure and fail to distinguish between some classes of phenetically different structures. The effectiveness of different dissimilarity measures when used by genetic operators will be examined in the subsequent chapter.

5 Enhanced Genetic Operators

5.1 Introduction

This chapter investigates the possibility of enhancing the native Framsticks' genetic operators by utilizing dissimilarity measures to introduce the desired neighborhood structure and preserve the features of parents in offspring solutions. The chapter is structured as follows. First, Sect. 5.2 assesses the locality of the native Framsticks' mutation and crossover operators using the `100_data_set`. Then, Sect. 5.3 presents the enhanced mutation and crossover operators that employ a dissimilarity measure to ensure a certain distance between a parent and an offspring solution. Next, a series of experiments examine the impact of the proposed genetic operators on the efficiency of the evolutionary search. Finally, Sect. 5.4 summarizes and discusses the main findings.

Due to the strong correlation between the `shape` and `dens` dissimilarity measures (shown in Chapter 3) and the high computational cost of the latter one, the `dens` measure is omitted from the subsequent analyses. Hence, the following four dissimilarity measures are employed in this chapter:

- `gene`,
- `opt`,
- `shape`,
- `freq`.

5.2 The locality of native operators

5.2.1 Mutation operator

The native mutation operators for *f0*, *f1*, and *f4*, which were introduced in Sect. 2.3, operate on a genotype by inserting, deleting, or altering a small part of it. Their aim is to produce a small change in the corresponding phenotype. The locality of these operators was examined using the structures from the `100_data_set`. For each representation and optimization objective (velocity on land, velocity in water, height of active structures, height of passive structures), 100 mutants were generated for every solution in the data

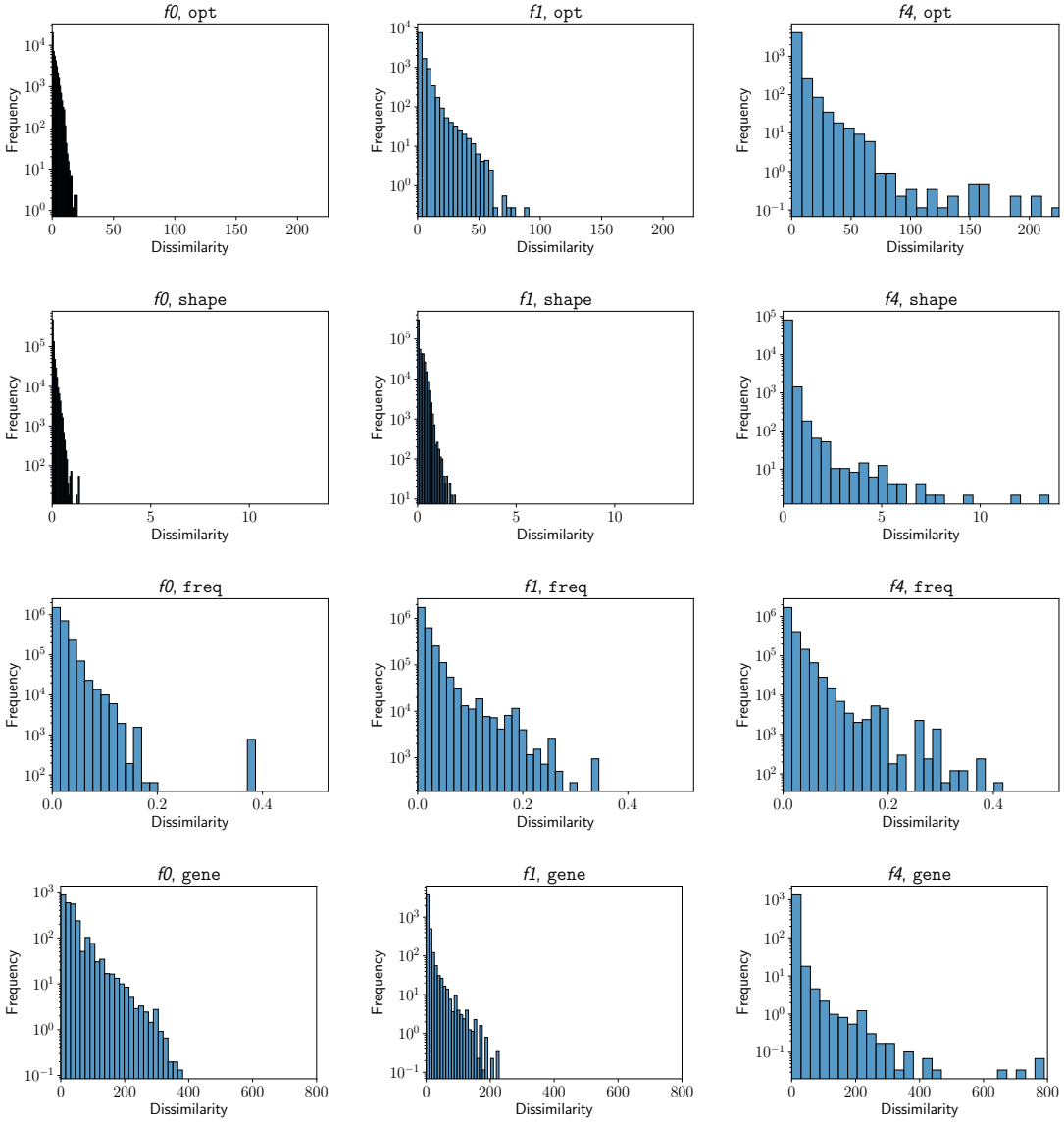


Figure 5.1: The distribution of the distance between the parent and mutant solutions, using different genetic encodings and dissimilarity measures. Each histogram is based on 400 solutions, 100 for each of the four considered optimization goals. The frequency is shown using a logarithmic scale. Rows: dissimilarity measures, columns: genetic encodings.

set. To evaluate the locality, the distance between a parent and each mutant solution was calculated, using each of the four dissimilarity metrics.

The results are presented in Fig. 5.1. In all cases, the distribution was right-skewed with the maximum in the first bin. This indicates that the mutations were most often small, neutral, or phenotypically neutral (when measured using the phenetic dissimilarity measures). The dissimilarity spread varied significantly across the genetic representations. The mutation operator of the low-level direct representation *f0* was the most local, resulting in the mutants that were the most similar to their parents. Conversely, the mutation operator for the developmental representation *f4* was the least local, and some of the mutants it produced were highly dissimilar from their parents. This result was consistent for all phenetic dissimilarity measures, and it could be attributed to the fact that the higher-level representations have more complex genotype-to-phenotype mappings. Thus for these representations applying a mutation operator to a genotype can potentially result in a larger change in a phenotype. The genotype-based `gene` was the only measure for which the *f1* mutation operator was more local than the *f0* operator because, in general, the genotypes for the *f0* are shorter and the employed Levenshtein distance was not normalized for length.

5.2.2 Crossover operator

The locality of the native crossover operators was evaluated by computing the distance between each parent pair and their offspring, using four dissimilarity measures. The structures from the `100_data_set` were used, which comprises 100 structures for each combination of the genetic encoding and the optimization goal. Each set of 100 structures was randomly partitioned into two subsets of 50 structures each, and crossover was applied to every possible pair of parents from the two subsets. This yielded 2500 offspring solutions for each setting.

The results are shown in Fig. 5.2. Similarly to the mutation operator, the crossover operator for the *f0* measure was the most local. The locality of the *f1* and *f4* representations was lower, resulting in a greater number of children that were highly dissimilar to one or both parents. Again, for the genetic `gene` measure, the locality was the highest for the *f1* representation that is characterized by shortest genotypes. Interestingly, for the `freq` measure, there were many more outliers for the *f1* encoding than for the *f4* encoding. A possible explanation for this could be that the developmental *f4* encoding supports modularity and regularity, and by swapping subtrees of parents, some regularity of parents is preserved in children, making them more similar to parents in terms of the frequency domain representation.

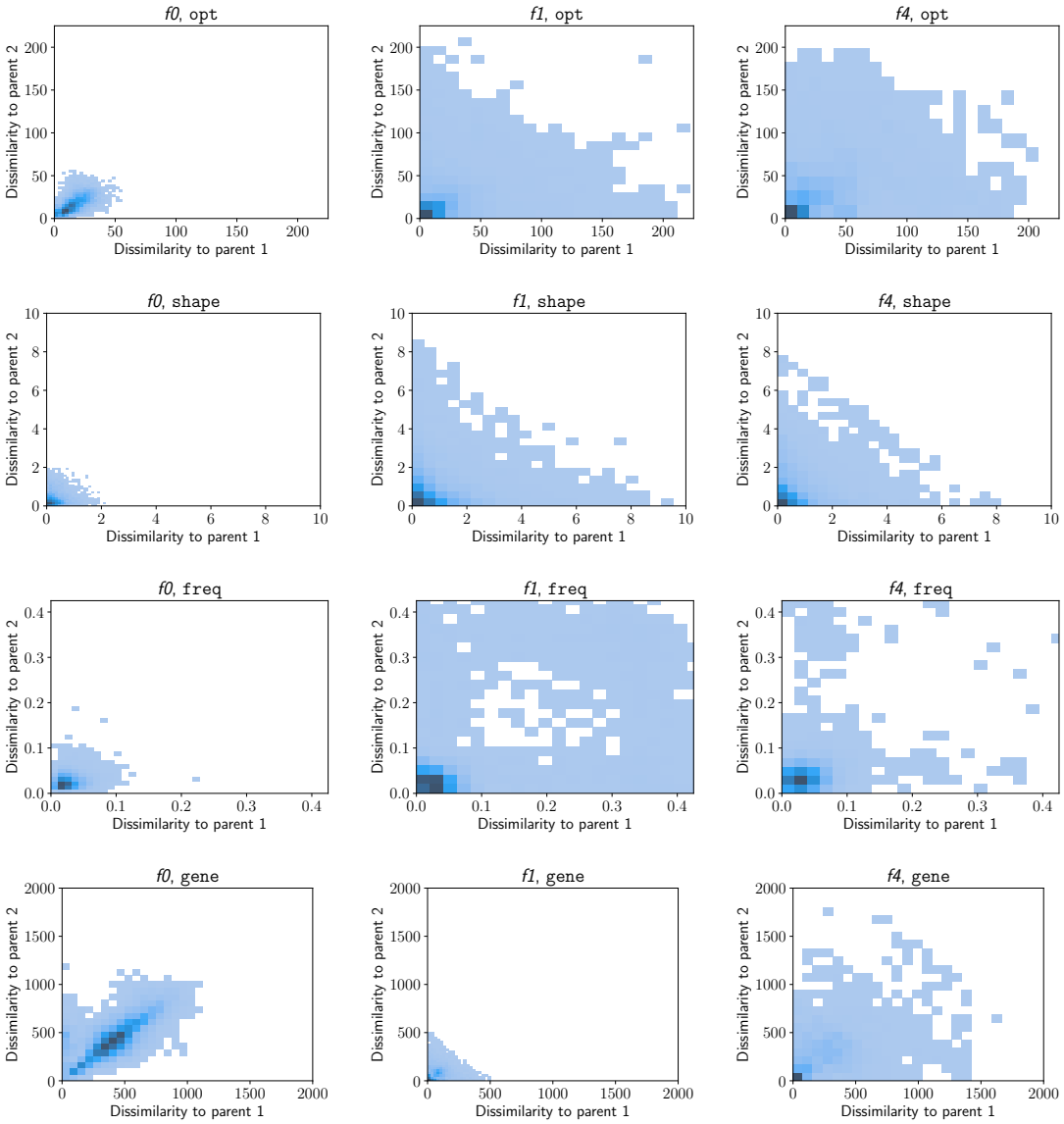


Figure 5.2: The distribution of the distance between two parents and a child solution, using different genetic encodings and dissimilarity measures. Each histogram is based on 400 parent solutions, 100 for each of the four considered optimization goals. The darker color indicates a higher frequency of observations. Rows: dissimilarity measures, columns: genetic encodings.

5.3 Enhanced genetic operators

5.3.1 Targeted sequential mutation operator

Definition

The Targeted Sequential Mutation (TSM) operator employs the native mutation operator and a specified dissimilarity measure to generate an offspring solution whose distance to the parent solution approximates a predefined value. The operator first creates a pool of k mutants in a sequential manner, using the native mutation operator. The initial mutant in the sequence is derived from the parent solution, and each subsequent mutant is derived from the preceding one. The operator then computes the dissimilarity between the parent solution and each of the k mutants in the pool, using the specified dissimilarity measure. The final offspring solution is selected as follows:

$$m^* = \underset{m \in m_1, m_2, \ldots, m_k}{\arg\min} |dissim(m, p) - target|,$$

where:

- m^* denotes the selected mutant solution,
- m_i denotes the i-th solution in the sequence generated by the native mutation operator,
- $dissim(m, p)$ denotes dissimilarity between a mutant and a parent solution,
- $target$ is the value of the TSM operator's `target` parameter.

This formula selects the mutant whose dissimilarity to the parent deviates the least from the `target` value. The parameter k, defining the number of mutants in the sequence, was set to 15, based on a trade-off between minimizing the deviation from the `target` value and assuring the computational efficiency.

The TSM operator aims to create a neighborhood structure according to the dissimilarity between the neighboring solutions. To this end, it uses the `target` parameter to maintain a specific distance from the parent solution, preventing the offspring solution from being either too dissimilar or too similar to the parent. The subsequent section explains in more detail how the `target` parameter value was experimentally determined for each dissimilarity measure. The operator generates mutants in a sequence rather than independently to enable the occurrence of neutral mutations (e.g. mutations of neurons, which are neutral according to purely phenotypic dissimilarity measures). The preliminary experiments showed that a version of this mutation operator that used an independently generated pool of mutants did not enhance the performance of the evolutionary in terms of best fitness.

Setting the `target` parameter value

The effectiveness of the TSM for different `target` values was investigated by conducting evolutionary experiments, which maximized the following objectives (in separate experiments):

Population size	50
Generations	4000
Selection method	Tournament
Tournament size	5
Mutation probability	0.65
Crossover probability	0
Maximum number of body parts	30
Maximum number of body joints	30
Maximum number of neurons	20
Maximum number of neuronal connections	30

Table 5.1: Parameters and constraints of the evolutionary algorithm used to investigate the effectiveness of the TSM operator and the influence of the `target` parameter value.

- velocity on land,
- height of passive structures.

The experimental parameters are shown in Table 5.1. In all experiments involving the enhanced mutation operator, the crossover was disabled. All three genetic representations were considered and the `opt` dissimilarity measure was employed in the TSM operator. The `opt` measure was chosen for this experiment as the most computationally efficient among the phenetic dissimilarity measures. Based on the analysis of the locality of the native mutation operator (Sect. 5.2), the following `target` values were selected: 4, 5, 6 ,7, 8, 10.

Fig. 5.3 shows the distribution of the most selected mutant index for each `target` value for both optimization objectives. For the *height passive* task, the first mutant in the sequence was, in general, the most frequently selected, which means that the first mutant's dissimilarity to the parent was closest to the desired value. This result may be explained by the passive character of this task. The neural network was not involved and only the morphology was affected by the mutations, therefore a lesser number of mutations was needed to reach the desired phenotypic outcome. For the *velocity on land* task, the most selected mutant index varied with the `target` value. For higher values of the parameter, more mutations were needed to obtain a mutant with a dissimilarity close to the `target`. The only exception was the *f4* encoding, which consistently showed the highest frequency of selection for the first mutant index, followed by a decreasing trend for subsequent indices. This result could be attributed to the low locality of the *f4* native mutation operator, which tends to generate more phenetically dissimilar mutants compared to the other two representations.

Fig. 5.4 displays the distribution of the best fitness value obtained in the evolutionary experiments for different optimization objectives, genetic representations, and `target` values using the native mutation operator and the TSM operator employing the `opt` measure. For the *height passive* objective, where the phenotype is most directly related to the fitness value, there is a trend for the median of the best fitness values to increase with the `target` value and then decrease after reaching a certain point. This result may suggest the existence of an optimal phenetic distance for solutions created by the mutation operator. For the *velocity on land* objective, where both the morphology and the neural network

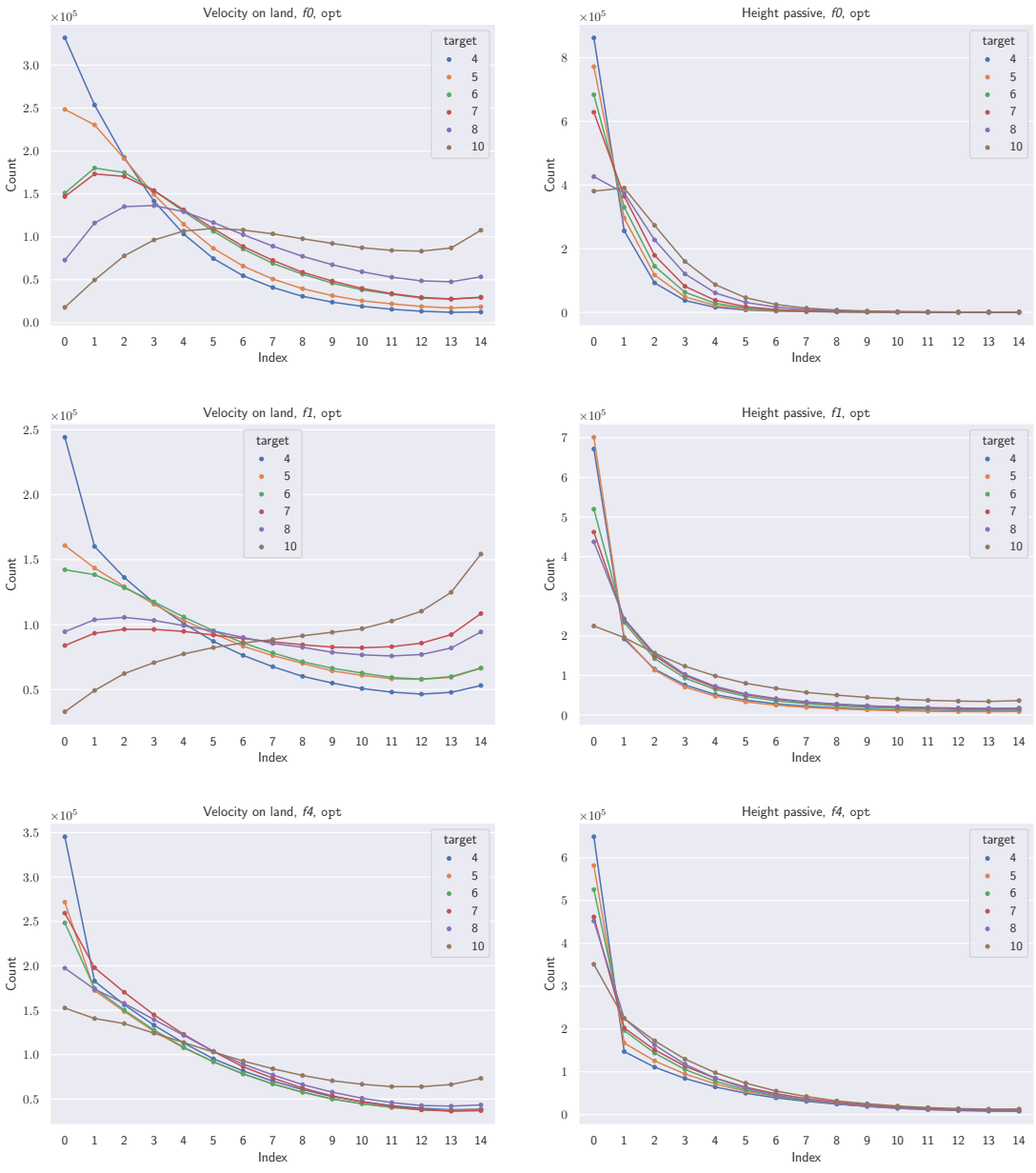


Figure 5.3: The number of the mutant selected by the TSM operator using the opt measure for different optimization goals and genetic representations. Rows: genetic encodings, columns: optimization objectives.

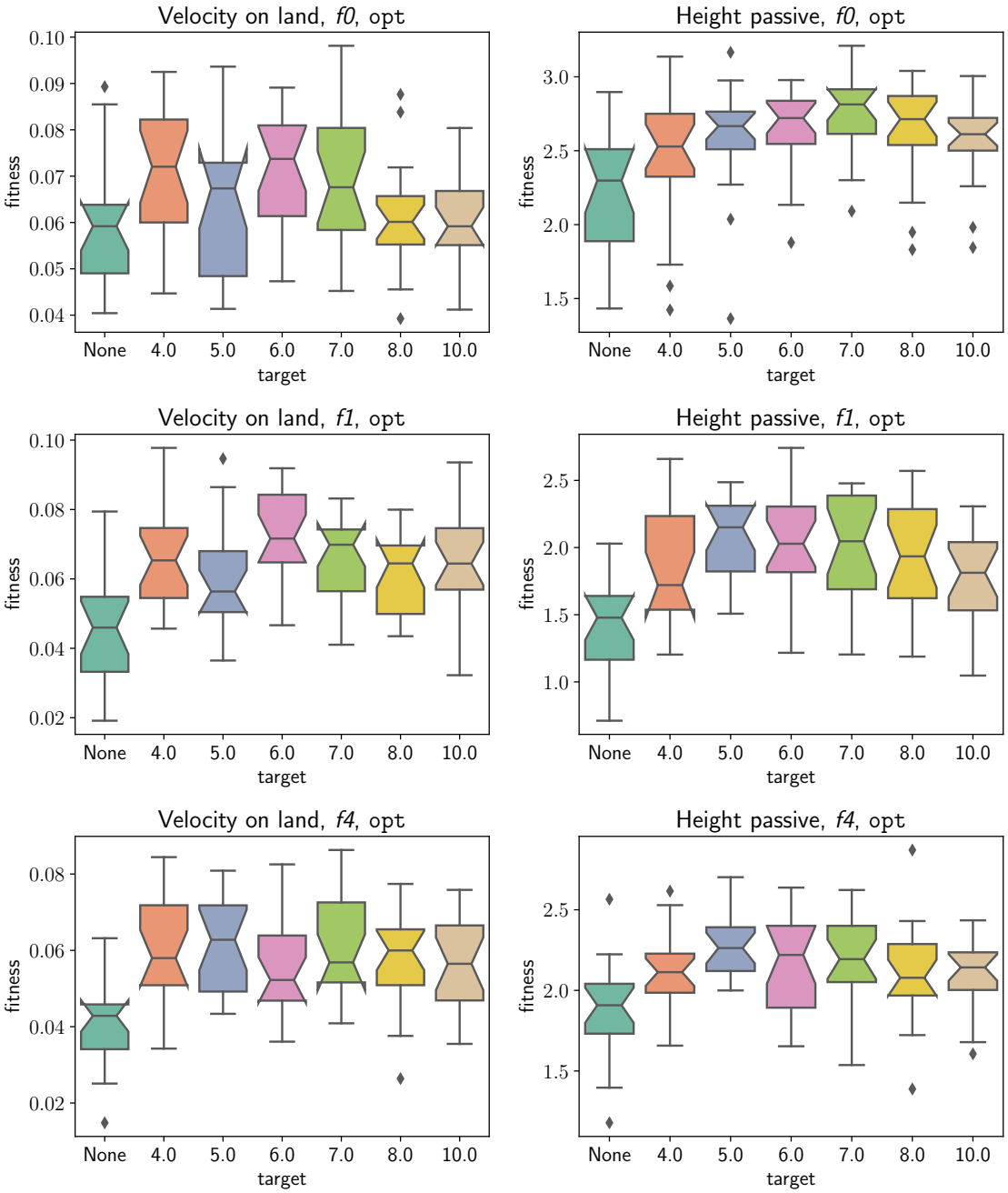


Figure 5.4: Fitness of the best individual for the TSM operator using the `opt` measure for different optimization goals, genetic representations, and `target` parameter values. The boxplots show the distribution of the best fitness values from 20 runs for each combination of the parameters.

contribute to the fitness value, this tendency is less evident. However, the TSM operator achieved a higher median of the best fitness values in comparison to the native operator for most `target` values. For both optimization goals, the greatest improvement of the best fitness median was observed, in general, for the `target` values 6 and 7.

The Kruskal-Wallis test [155] was conducted to test the statistical difference between the medians of the best fitness obtained using the native mutation operator and the TSM operator with different targets. For each combination of optimization goal and genetic representation the following hypotheses were tested:

H_0: The medians of the best fitness obtained using the native mutation operator and the TSM operator are equal.

H_1 : The medians of the best fitness obtained using the native mutation operator and the TSM operator differ significantly for at least one `target` value.

The significance level was set at 0.05. Dunn's post-hoc test was conducted for the cases where the null hypotheses were rejected. Table 5.2 presents the results. In all cases except the *velocity on land* task using the *f0* representation, there were `target` values for which the median of best fitness differed significantly from the median for the native operator. In most of those cases, the `target` values that yielded significantly different medians were 6.0, 7.0 (5 out of 5 each), and 8.0 (3 out of 5).

Search characteristics comparison

Fig. 5.5 compares the fitness of the best individual across ten sample runs for the native mutation operator and the TSM mutation operator using the `target` value that yielded the highest median of the best fitness. In the *velocity on land* task, the native mutation operator achieves a rapid improvement in fitness, within the first few hundred generations. After that, there is little or no improvement in most cases, indicating that the algorithm has fallen into a local optimum. On the other hand, the TSM operator shows a gradual increase in fitness, ultimately attaining higher fitness values and exhibiting potential for further improvement. This result suggests that the TSM operator defines a neighborhood structure that facilitates a more effective exploration of the solution space. For the *height passive* task, the fitness of the best individual for the native mutation operator converges in most cases before the 1000-th generation. The TSM operator, again, exhibits a more

Goal	Representation	H	p-value	Significant targets*
Velocity on land	*f1*	29.932	0.000	4.0, 6.0, 7.0, 10.0
	f4	27.452	0.000	4.0, 5.0, 6.0, 7.0, 8.0, 10.0
Height passive	*f0*	20.878	0.002	6.0, 7.0, 8.0
	f1	29.514	0.000	5.0, 6.0, 7.0, 8.0
	f4	22.306	0.001	5.0, 6.0, 7.0

Table 5.2: Results of the Kruskal-Wallis test and Dunn's test with Bonferroni's correction for comparing the best fitness values obtained using the native mutation operator and the TSM operator with different targets. For each considered optimization goal and genetic representation the H-statistic, the p-value, and the values of targets for which best fitness differed significantly from that obtained using the native mutation operator are shown. *The significance level was set at 0.05.

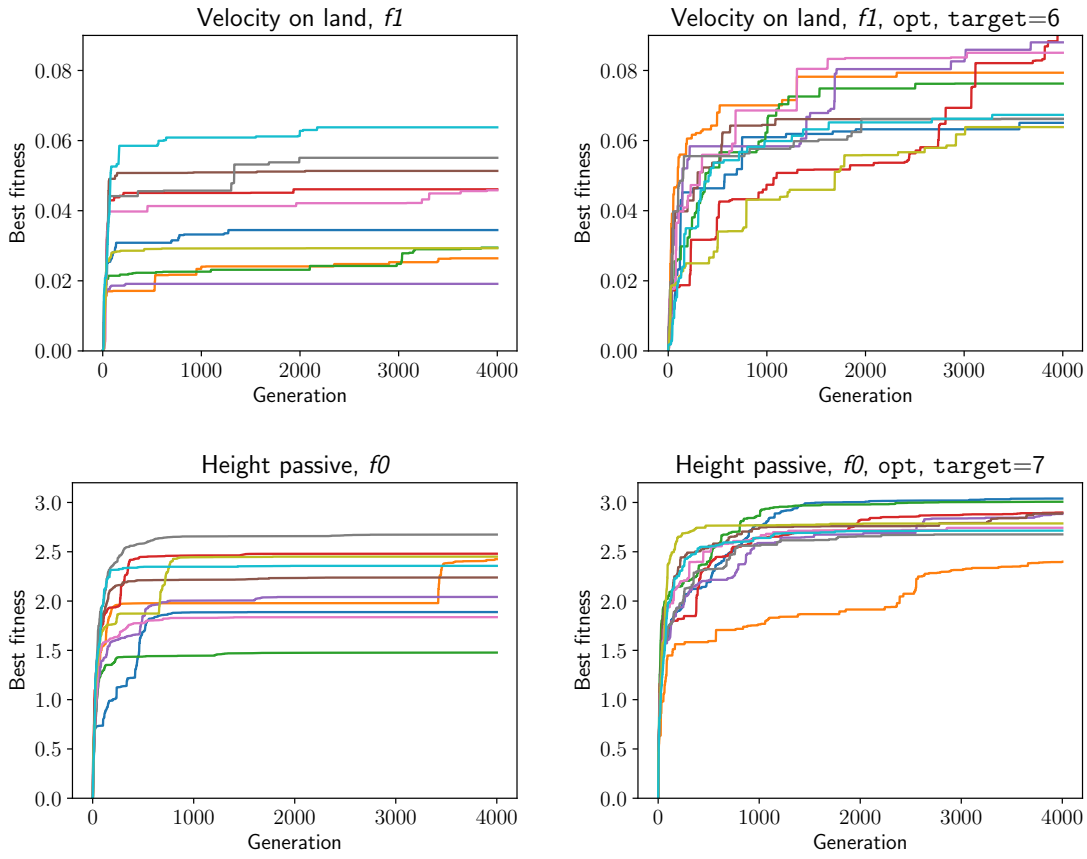


Figure 5.5: Fitness of the best individual in each generation for *velocity on land* and *height passive*, using genetic encodings *f0* and *f1*, respectively. The left column shows the results for the native mutation operator and the right column shows the results for the TSM operator using the `opt` measure and the best `target` value. Colors correspond to different runs of the algorithm.

gradual increase in the fitness of the best individual, achieving higher final fitness values than the native operator.

Dissimilarity measures comparison

The performance of the TSM operator using different dissimilarity measures was compared across all genetic representations and optimization goals in evolutionary experiments. The experimental parameters were the same as in the `target` setting experiment (see Table 5.1). However, due to the higher computational cost of measures other than the `opt` measure, the value of the k parameter was reduced to 10 for them. Also, the parameter values of the measures were decreased as follows:

- `shape`:
  - `density`: 5.0,
  - `bins number`: 128,
  - `samples number`: 1000.

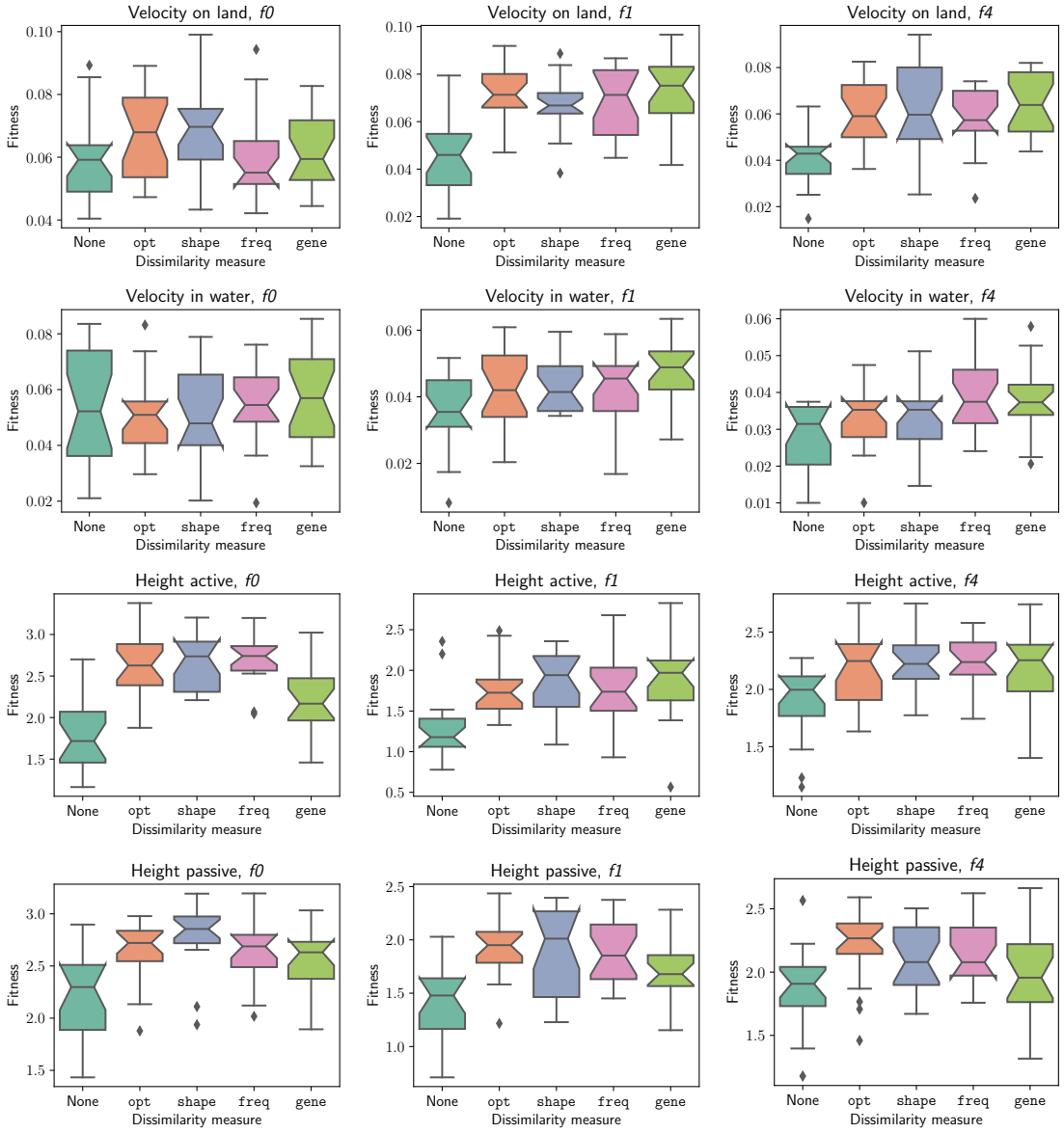


Figure 5.6: Fitness of the best individual for all optimization goals and genetic representation for the native mutation operator and the TSM operator employing different dissimilarity measures. The boxplots show the distribution of the best fitness values from 20 runs for each combination of parameters. Rows: optimization objectives, columns: genetic encodings.

- `freq`:
  - `sampling density`: 5.0,
  - `resolution` 5.0.

The measures used the following `target` values, which were determined experimentally to maximize best fitness:

- `opt`: 6.0,
- `shape`: 0.15,
- `freq`: 0.07,
- `gene`: 32.5.

The Kruskal-Wallis test was conducted to test the statistical difference between medians of the best fitness obtained using the native mutation operator and the TSM operator employing different dissimilarity measures. For each combination of optimization objective and genetic representation, the test evaluated the following hypotheses:

H_0: The medians of the best fitness obtained using the native mutation operator and the TSM operator are equal.

H_1 : The medians of the best fitness obtained using the native mutation operator and the TSM operator differ significantly for at least one dissimilarity measure.

The significance level was set at 0.05. Dunn's post-hoc test with Bonferroni's correction was performed for the cases where the null hypotheses were rejected.

Goal	Representation	H	p-value	Significant measures*
Velocity on land	*f1*	30.375	0.000	`opt`, `shape`, `freq`, `gene`
	f4	27.466	0.000	`opt`, `shape`, `freq`, `gene`
Velocity in water	*f1*	9.731	0.045	`gene`
	f4	13.165	0.010	`freq`, `gene`
Height active	*f0*	45.772	0.000	`opt`, `shape`, `freq`
	f1	24.957	0.000	`opt`, `shape`, `freq`, `gene`
	f4	15.816	0.003	`shape`, `freq`, `gene`
Height passive	*f0*	24.861	0.000	`opt`, `shape`
	f1	21.158	0.000	`opt`, `shape`, `freq`
	f4	14.154	0.007	`opt`

Table 5.3: Results of the Kruskal-Wallis test and Dunn's post-hoc test with Bonferroni's correction for the TSM operator using different dissimilarity measures compared to the native operator. For each considered optimization goal and genetic representation the H-statistic, the p-value, and the dissimilarity measures for which best fitness median differed significantly from that obtained using the native mutation operator are shown. *The significance level was set at 0.05.

Table 5.3 reports the results of the Kruskal-Wallis test and Dunn's post-hoc test. Fig. 5.6 shows the distribution of fitness of the best individual for each genetic representation and optimization goal, for the native mutation operator and the TSM operator using different dissimilarity measures. The application of the TSM operator improved the best fitness for all optimization goals and genetic encodings, except for the velocity-related

tasks using the *f0* genetic representation. The effectiveness of the different dissimilarity measures varied depending on the genetic encoding and task. Among the four optimization goals, the *velocity in water* task showed the least number of cases where the TSM operator significantly improved the best fitness. A possible reason for this outcome is that the fitness landscape for this problem has different characteristics, and the `target` value established based on the *velocity on land* and *height passive* tasks is not optimal in this case. Interestingly, the measure that improved the best fitness in this task for both *f1* and *f4* was the genotype-based `gene` measure.

None of the measures employed in the TSM operator clearly outperformed the other measures. For all optimization objectives and genetic encodings, the significant improvements were achieved equally often using the `opt`, `shape` and `freq` measures (7 out of 9 cases each). However, it is noteworthy that the `gene` measure was also effective in 6 cases, all involving active structures, and for *velocity in water* objective and the *f1* encoding it was the only measure which yielded improvement. The results for `gene` measure – which also considers the neural network encoded in the genotype – may imply that for active structures it would be beneficial to incorporate the neural distance into dissimilarity assessment.

Computational cost

The most computationally intensive step of the TSM operator was calculating the dissimilarity between the solutions. Thus, the execution time of a single evolutionary run mainly depended on the value of the k parameter (the size of the mutants pool) and the chosen dissimilarity measure, as they had different computational costs (see Sect. 3.8.1). The time of a single run on a single core of an Intel Xeon Gold 5320 2.20GHz CPU for the considered phenetic measures spanned from on average 8 hours (for the `opt` measure with $k = 15$) to on average 122 hours (for the `freq` measure with $k = 10$). For the standard mutation operator time of a single run was on average 2 hours.

5.3.2 Enhanced crossover operators

Four variants of the enhanced crossover operator were implemented and tested. Each of those operators utilizes a dissimilarity measure to preserve the phenetic traits of parents in a child solution. Their performance was compared to the performance of the native crossover operator in the selected optimization problems. Then, the locality of the introduced crossover operators was analyzed and compared to that of the native operator.

Distance Preserving Crossover (DPX)

The Distance Preserving Crossover (DPX) developed in this work is based on the DPX operator introduced by Merz and Freisleben [61]. Its aim is to produce offspring that are not more dissimilar to either of the parents than the parents are to each other. To achieve this goal, a pool of k offspring is created using the native crossover operator and the dissimilarity between both parents and each of k offspring is computed using a selected dissimilarity measure. The final offspring solution is selected as follows:

$$c^* = \begin{cases} \text{random}(C') & \text{if } C' \neq \emptyset \\ \text{random}(C) & \text{if } C' = \emptyset \end{cases}$$

where:

- c^* denotes the selected child solution,
- $C = c_1, c_2, ..., c_k$ is the pool of k mutants created using the native crossover operator,
- $C' = c \in C : dissim(c, p_1) \leq dissim(p_1, p_2) \wedge dissim(c, p_2) \leq dissim(p_1, p_2)$ is the subset of C that satisfies the DPX condition.

The parameter k was set to 5. The preliminary experiments (for this and the following operators which also use the pool of k mutants) showed that increasing the pool size to 10 does not significantly increase the best fitness.

Distance Minimizing Crossover (DMX)

The Distance Minimizing Crossover (DMX) follows a similar idea as the approximate geometric crossover proposed by Krawiec and Lichocki for genetic programming [129]. It aims to produce offspring that are phenotypically (or genotypically when using the genotype-based dissimilarity measure) intermediate between the parents by minimizing the distance between the child solution and parent solutions. To achieve this, a pool of k offspring is generated using the native crossover operator and the dissimilarity between each of k offspring and both parents is computed using a selected dissimilarity measure. The final child solution is chosen from the k candidates as follows:

$$c^* = \underset{c \in c_1, c_2, ..., c_k}{\arg\min} \ (dissim(c, p_1) + dissim(c, p_2)),$$

where:

- c^* denotes the selected child solution,
- c_1 denotes the i-th solution generated by the crossover operator from parent p_1 and p_2,
- $dissim(c, p_i)$ denotes the dissimilarity between the child and the i-th parent.

This formula selects the child solution that has the smallest average dissimilarity to both parents, based on the selected dissimilarity measure. The parameter k was set to 5.

Equidistance Minimizing Crossover (EMX)

The DMX crossover operator presented in the previous section may produce child solutions that are identical to one of the parents. To prevent this effect, the formula of the DMX operator was modified by adding a term that penalizes the unequal distance to the parents, following the idea from [129]:

$$c^* = \underset{c \in c_1, c_2, ..., c_k}{\arg\min} \ (dissim(c, p_1) + dissim(c, p_2) + |dissim(c, p_1) - dissim(c, p_2)|).$$

The added term promotes solutions that are equally or nearly equally distant from both parents, since its value is zero or close to zero for those solutions.

Similarity-Based Crossover (SBX)

Unlike the three new crossover operators that were based on computing the dissimilarity between a pool of k mutants generated by the native mutation operator and both parents, the Similarity-Based Crossover (SBX) adopts a different approach to control the phenotypic similarity between parent and offspring solutions. The SBX selects the second parent for the crossover according to its similarity to the first parent, measured by a chosen dissimilarity measure. The selection of the second parent is performed as follows:

$$p_2 = \underset{p \in P \setminus \{p_1\}}{\arg\min} (dissim(p, p_1)),$$

where:

- p_1 denotes the first parent selected for the crossover,
- p_2 denotes the second parent selected for the crossover,
- P denotes the current population,
- $dissim(p, p_1)$ denotes dissimilarity between the first parent solution and a solution p.

Therefore, the SBX aims to create offspring solutions that share phenotypic traits with both parents by crossing over the solutions that are most similar to each other.

The crossover operators performance comparison

Population size	50
Generations	4000
Selection method	Tournament
Tournament size	5
Mutation probability	0.65
Crossover probability	0.25
Maximum number of body parts	30
Maximum number of body joints	30
Maximum number of neurons	20
Maximum number of neuronal connections	30

Table 5.4: Parameters and constraints of the evolutionary algorithm used to compare the performance of the proposed crossover operators.

Similar to the enhanced mutation operator, the effectiveness of the four proposed crossover operators was evaluated by conducting evolutionary experiments, which optimized the following objectives (in separate experiments):

- velocity on land,
- height of passive structures.

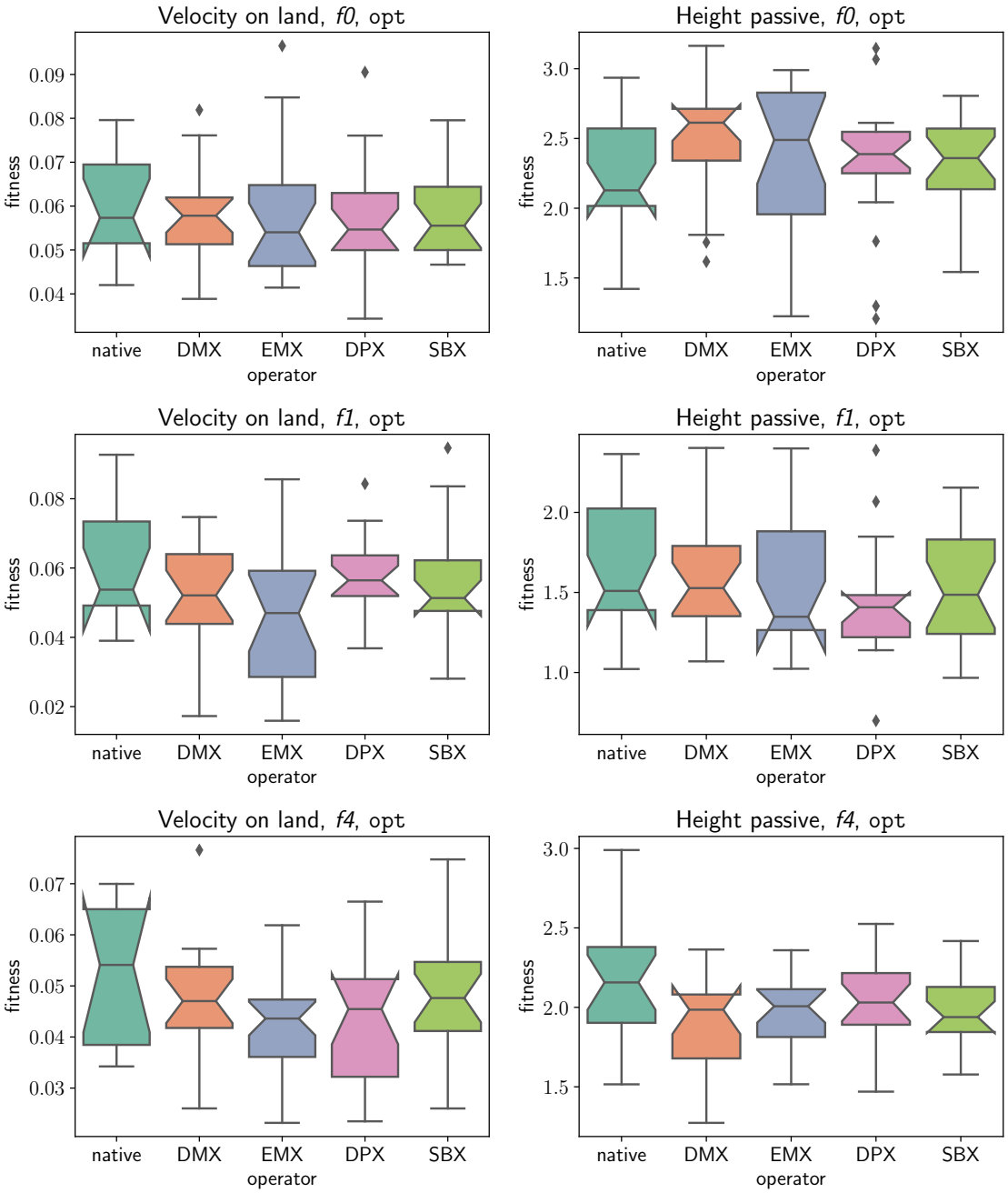


Figure 5.7: Fitness of the best found solutions for the native crossover operator and DMX, EMX, DPX, and SBX operators using the opt measure for different optimization goals, and genetic representations. The boxplots show the distribution of the best fitness values from 20 runs for each combination of parameters. Rows: genetic encodings, columns: optimization objectives.

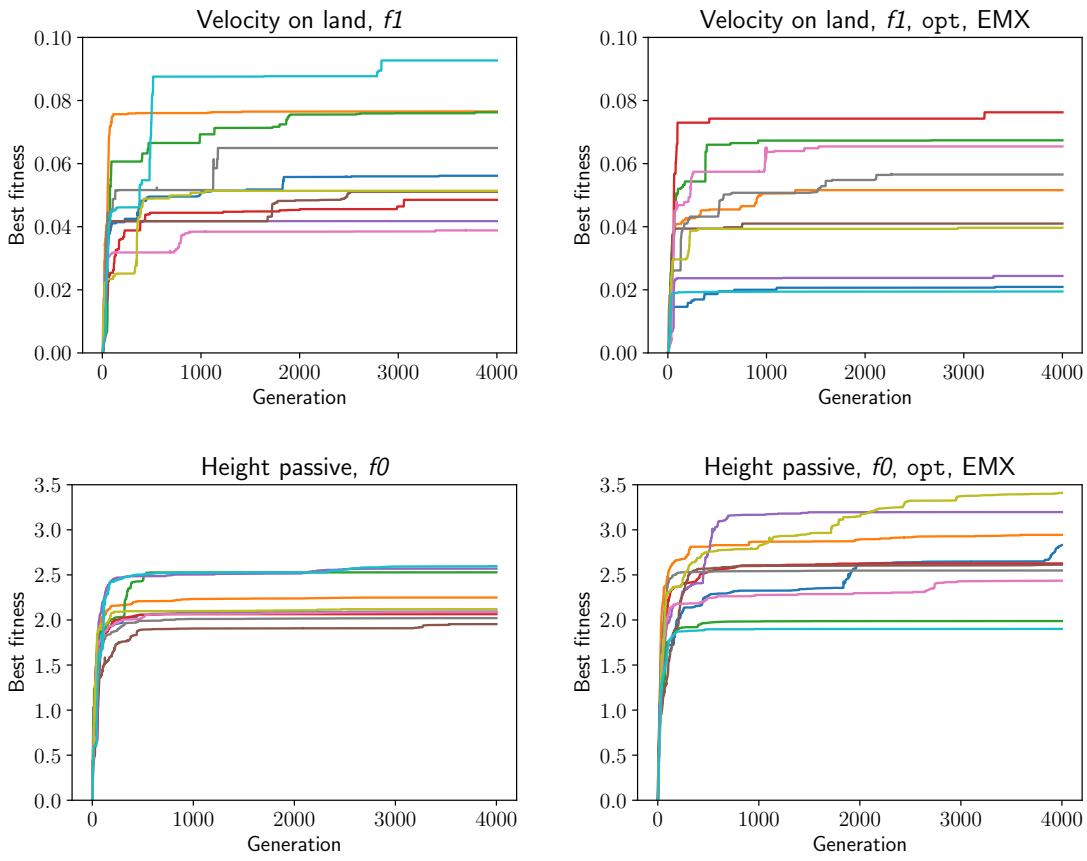


Figure 5.8: Fitness of the best individual in each generation for the *velocity on land* and *height passive* task, using the genetic encodings *f0* and *f1*. The left column show the results for the native crossover operator and the right column shows the results for the EMX operator. The colors represent different independent runs of the algorithm.

Table 5.4 presents the experimental parameters used for the comparison of the crossover operators. The native mutation operator was used. The `freq` measure, which was the most computationally intensive, was excluded from the analysis, as the preliminary experiments showed that it did not yield better results. Therefore only the measures `gene`, `opt`, and `shape` were considered. Fig. 5.7 shows the distribution of the best fitness for the crossover operators using the `opt` measure, and Figs. G.1–G.2 in Appendix G show the same for the crossover operators using `shape` and `gene` measures, respectively. The results indicate that DMX, EMX, DPX, and SBX perform similarly to the native crossover operator. None of the enhanced crossover operators achieved a significant improvement in terms of the best fitness. The comparison of the best fitness in subsequent generations between the native operator and the enhanced operators revealed that the search process characteristics were similar for all five operators, especially for the *velocity on land* task. Fig. 5.8 illustrates a sample of evolutionary runs for the native and the EMX operators. For the *height passive* task, the enhanced crossover operator exhibited a more gradual increase in fitness of the best solution, but the final results were not significantly better than those of the native operator.

Locality

The locality of the enhanced crossover operators was evaluated and compared to that of the native operator. The same method as for the native operator (see Sect. 5.2.2) was used to assess the locality, using the `100_data_set`. The SBX operator was omitted from the analysis as it is based on selecting the most similar second parent from the current population. Fig. 5.9 shows the results for the `opt` measure. As expected, the DMX operator produces more offspring solutions that are highly similar to one of the parents, especially for the *f0* and *f4* encodings. The EMX and DPX operators, which is also expected, produce a higher number of children with similar distance to both parents. The results also indicate that the DMX and EMX operators produce fewer offspring solutions that are highly dissimilar to both parents than the native operator. However, these differences are not substantial, and the overall distribution of dissimilarities for the offspring solutions is similar across all operators. This indicates that the enhanced crossover operators did not perform significantly better than the native crossover operator in terms of the locality. All the enhanced operators, however, depended on the native one, which may limit their potential for improvement.

Computational cost

As in the case of the TSM operator, the most computationally intensive step of the crossover operator was calculating the dissimilarity between the solutions. Therefore, the execution time of a single evolutionary run depended on the number of solutions for which the distance was calculated and on the selected measure. DPX, DMX, and EMX computed the distance between each parent and five candidate children solutions. SBX operator required computation of the dissimilarity matrix for the entire population to select the second parent that was the most similar to the first parent, and thus was much more computationally intensive. The time of a single run on a single core of an Intel Xeon Gold 5320 2.20GHz CPU for the considered phenetic measures ranged from on average three hours (for the DPX, DMX, and EMX operators using the `opt` measure) to on average 45 hours (for the SBX operator using the `shape` measure). For the standard mutation operator time of a single run was on average 2 hours.

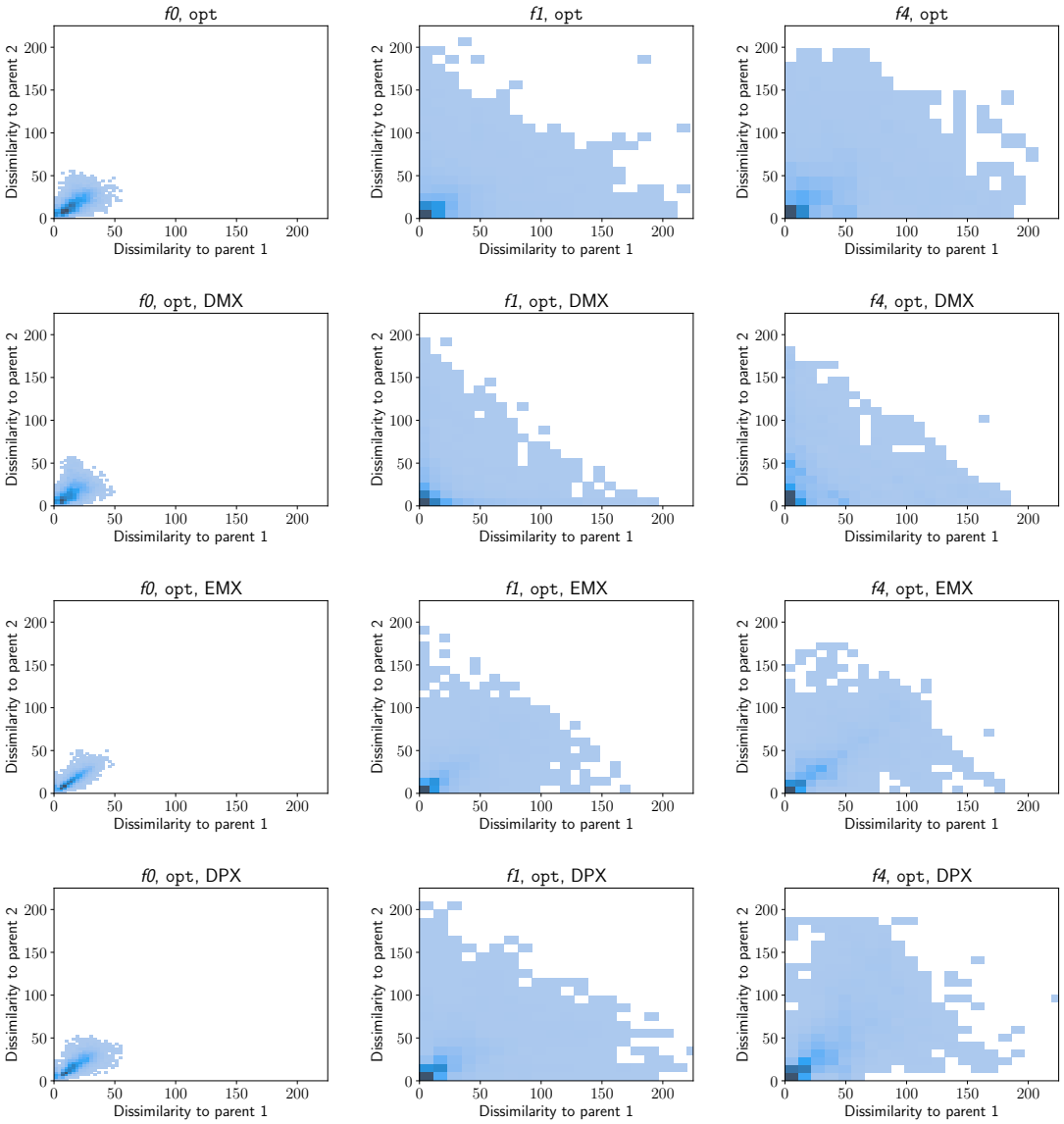


Figure 5.9: The distribution of the distance between two parents and a child solution, using different genetic encodings and the `opt` measure. Each histogram is based on 400 solutions, 100 for each of the four considered optimization goals. The darker color indicates a higher frequency of observations. Rows: crossover operators (native, DMX, EMX, DPX). Columns: genetic encodings (*f0*, *f1*, *f4*).

5.4 Summary

This chapter analyzed the locality of the native Framsticks genetic operators and proposed genetic operators that utilize dissimilarity measures for 3D structures. The locality of the native mutation and crossover operators varied significantly across the considered genetic representations, with the low-level *f0* representation being the most local and the high-level developmental representation *f4* being the least local. The proposed TSM mutation operator, which aims to ensure a certain distance of a mutant to its parent, improved the performance of the evolutionary search in terms of the best fitness compared to the native mutation operator for most representations, in all of the considered optimization tasks, especially those involving maximizing the height of the structures. The influence of its `target` parameter value on the performance of the algorithm in terms of the best fitness indicated the existence of a potential optimal `target` value for defining a neighborhood structure using a given dissimilarity measure. The lower performance of the TSM operator in *velocity in water* task may suggest that the optimal `target` is different for different optimization objectives, because of different fitness landscapes, and that it should be adjusted independently for each of those objectives. The good performance of the `gene` measure for this objective may also imply that for active designs, a more sophisticated measure of neural dissimilarity than the one currently used in the `opt` measure is needed, such as [122].

Unlike the mutation operator, none of the newly developed crossover operators outperformed the native crossover operator. The new operators achieved similar performance to the native operator in terms of the best fitness. Furthermore, the locality analysis and the comparison of the best fitness in subsequent generations for the native and new operators revealed that they had similar characteristics. Thus, the hypothesis that using a dissimilarity measure with high FDC to preserve the parent features in offspring solutions would improve the search efficiency was not supported for the crossover operator. However, it should be noted that all the proposed operators depended on the native crossover operator to generate offspring solutions, which could constrain their potential to surpass it. A possible direction for further improvement of the crossover operator is to develop a distance-based operator that does not rely on the native operator.

6

Dissimilarity Measures in Diversity Maintenance Techniques

6.1 Introduction

The previous chapters examined the global convexity of fitness landscapes in evolutionary design (ED) problems and investigated the potential of using dissimilarity measures to design enhanced genetic operators. Another way to utilize dissimilarity measures to improve the performance of evolutionary search in ED problems is to apply them in diversity maintenance techniques, which are methods that aim to prevent premature convergence and preserve population diversity in the presence of rugged, multimodal fitness landscapes.

There exist various explicit methods to maintain diversity and increase the exploration of the solution space, however, the most common techniques are based on niching [190]. Another, more radical, approach to encourage the exploration is to disregard the fitness function and assess solutions solely by their novelty. This work compares both of these approaches using the developed dissimilarity measures. The approaches are discussed in more detail below.

Niching techniques derive from the biological concept of niches. Living organisms inhabiting a niche compete for resources among themselves, but not with organisms from the other niches. Analogously, in optimization algorithms, a niche is a subregion of the solution space. The main goal of diversity maintenance techniques is to find and preserve solutions in various niches. Fitness sharing [82, 70, 143] is one of the most prevalent niching methods in evolutionary computation. It reduces the fitness value of an individual solution according to its similarity to other solutions in the population, to promote solutions that occupy underexplored regions of the solution space. The similarity between solutions can be assessed by genetic or phenetic characteristics, using various distance measures. This technique has been effective in many practical problems such as antenna design [134] and battery charging/swap station design [230]. Other common niching techniques are clearing, crowding and clustering [67].

Lehman and Stanley [139, 140] proposed novelty search as a method of addressing the challenge of convergence to local optima and dealing with deceptive fitness landscapes. The main idea of novelty search is to discard the fitness function as an evaluation criterion and use the novelty metric instead, usually measuring the novelty of the phenotype or

the behavior. Thus, the algorithm performs an open-ended search without an explicit optimization goal. Novelty search has been effective in challenging optimization tasks such as artificial neural network evolution [184], body-brain co-evolution [131], and genetic programming [54].

Novelty search with local competition (NSLC) is a variant of novelty search that introduces competition among solutions within niches. It was first proposed by Lehman and Stanley [141] to evolve morphologies of artificial creatures. They found that NSLC, compared to a simple novelty search where every solution competed with every other solution, produced morphologies that were both functional and diverse. NSLC belongs to the class of quality-diversity algorithms [175], which balance the optimization of the fitness function with the diversity of solutions. These algorithms are not considered in this work, as its goal is to first analyze the influence of the chosen dissimilarity measure using the more basic diversity maintenance methods. However, they will be investigated in future work.

This chapter performs an extensive comparison of diversity maintenance techniques, focusing on the impact of the selected distance measures on the outcomes of the evolutionary optimization process. It is a follow-up to the research published in [114]. The chapter is organized into sections detailing the diversity maintenance algorithms used, experimental parameters and results, and a summary of the main findings.

6.2 Methods

The experiments compared six types of evolutionary algorithms:

- a standard evolutionary algorithm without diversity maintenance,
- niching (with local and global variants),
- novelty search (with local and global variants),
- the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [48].

The niching and novelty search algorithms measured the distance to either k nearest neighbors or all solutions in the current population and the archive, depending on the variant. NSGA-II maximized fitness and distance as its two objectives.

6.2.1 Solutions archive

The solutions archive was used in niching and novelty search to keep track of the regions of the solution space visited so far. It had a fixed size of 50 solutions and was updated in every generation of the algorithm. When a new individual was added to the archive, one solution from the archive was removed if the archive size was exceeded. The removal criterion was the lowest dissimilarity score, which indicated the most similar pair of solutions in the archive. To maintain archive stability, the last added solution was removed if it belonged to the most similar pair; otherwise, a random solution from the pair was removed. This policy aimed to prevent the archive from being dominated by a single solution or a cluster of similar solutions. Listing 6.1 outlines the algorithm for managing the solutions archive.

```
# A method to remove an individual from the archive based on the dissimilarity
def remove_individual(dissim):
  # Find the pair of individuals that are most similar in the archive
  most_similar_pair = find_most_similar_pair(dissim)

  # If the last individual in the archive is one of them, remove it
  if last_individual in most_similar_pair:
    return last_individual

  # Otherwise, remove a random individual from the pair
  else:
    return random_choice(most_similar_pair)

# A method to update the archive with the current population
def update_archive(dissim_matrix, population_archive):
  #If the archive size is zero, do nothing
    if archive_size < 1:
      return

  # Get the indices of solutions in current archive
  current_archive_ind = get_current_archive_ind(dissim_matrix)

  # Loop over each individual in the population
  for i in range(population_size):
    # Add the individual's index to the current archive indices
    current_archive_ind.append(i)

    # Get a submatrix of the dissimilarity matrix with the current archive
    indices
    temp_dissim_matrix = get_submatrix(dissim_matrix, current_archive_ind)

    # If the current archive size exceeds the limit, remove an individual
    if len(current_archive_ind) > archive_size:
      to_remove = remove_individual(temp_dissim_matrix)
      current_archive_ind.pop(to_remove)

  # Update the archive with the individuals from the population archive
  archive = get_individuals(population_archive, current_archive_ind)
```

Listing 6.1: The algorithm for managing the solutions archive.

6.2.2 Niching

The main difference between niching and the standard evolutionary algorithm was that niching used an archive to store and retrieve diverse solutions, and that it modified the fitness value of each solution (hereinafter referred to as raw fitness) by multiplying it by a diversity score. This way, niching encouraged the exploration of different regions of the solution space, rather than converging to a single optimum.

The global version of niching calculated the fitness of the i-th solution as follows:

$$fitness_i = raw\_fitness_i \cdot distance\_global_i \tag{6.1}$$

with diversity score $distance\_global_i$ calculated as:

$$distance\_global_i = \frac{1}{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} dist(x_i, x_j) \tag{6.2}$$

where n denotes the combined size of the population and the archive, and $dist$ is the distance (dissimilarity) measure.

In the local version of niching, the fitness of the i-th solution was calculated as:

$$fitness_i = raw\_fitness_i \cdot distance\_local_i \tag{6.3}$$

with diversity score $distance\_local_i$ calculated as:

$$distance\_local_i = \frac{1}{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} dist(x_i, x_j) \tag{6.4}$$

where k is the parameter determining the number of solutions closest to the i-th solution according to the distance (dissimilarity) measure.

6.2.3 Novelty search

Novelty search differed from the standard evolutionary algorithm in that it used the archive to store and retrieve diverse solutions, and that it replaced the raw fitness of each solution with its diversity score. Thus, the only objective that the novelty search was guided by was the phenotypic (or genetic if the `gene` measure was used) diversity of the solutions. In the global version, the diversity score was calculated as the distance to the current population and solutions archive using (6.2). In the local version it was calculated as the distance to k-nearest neighbors using (6.4).

6.3 Experiments and results

6.3.1 Experimental parameters

The preliminary experiments [114] revealed that the choice of genetic encoding did not affect the relative performance of different diversity maintenance techniques. Hence, this

work focuses on comparing diversity maintenance techniques combined with different dissimilarity measures employing only the *f1* genetic representation. Two optimization tasks were selected, involving active and passive structures, respectively:

- velocity on land,
- height of passive structures.

Five diversity maintenance techniques for evolutionary algorithms were compared with a standard EA without any diversity maintenance, which served as the baseline. The compared algorithms are listed below together with their abbreviations used hereinafter.

- Standard EA without any diversity maintenance, the baseline (*none*).
- Niching using the distance to k-nearest neighbors (*niching local*).
- Niching using the distance to the entire population and archive (*niching global*).
- Novelty search using the distance to k-nearest neighbors (*novelty local*).
- Novelty using the distance to the entire population and archive (*novelty global*).
- Non-dominated Sorting Genetic Algorithm II separately maximizing the fitness and novelty objectives (*NSGA-II*).

For each diversity maintenance technique, three different dissimilarity measures were used separately to evaluate the distance between solutions. The dissimilarity measures and their parameter values were:

- `gene`
- `opt`:

  $w_V = 0$

  $w_D = 1$

  $w_N = 0.1$

  $w_G = 1$

- `shape`

  `descriptor`: `d2`

  `density`: 5.0

  `bins number`: 128

  `samples number`: 1000

For the `opt` measure the weights of the components that correspond to the similarity in terms of matched vertices, w_D and w_G were set to 1. To account for the neural distance but also to prevent this component from dominating the aggregated dissimilarity value, its weight was set *to*0.1. The parameter values for the `shape` measure were set based on the results of the tuning procedure (Sect. 3.5). The value of `samples number` was decreased to 1000 to increase the computational efficiency.

The evolutionary algorithm used in this study was tuned based on several preliminary experiments to ensure convergence. The parameters of the algorithm are shown in Table 6.1. To value of the parameter k, determining the number of nearest neighbors used

Population size	100
Generations	3000
Selection method	Tournament
Tournament size	5
Mutation probability	0.7
Crossover probability	0.2
Archive size	50
Maximum number of body parts	15
Maximum number of body joints	30
Maximum number of neurons	15
Maximum number of neuronal connections	30

Table 6.1: Parameters and constraints of the evolutionary algorithm used to compare the performance of the diversity maintenance techniques.

for local novelty search and niching, was set to value 5. Preliminary experiments with values of 2 and 30 showed no significant difference in the results. The native mutation and crossover operators were used.

The following settings were examined: two optimization objectives, one baseline and five diversity maintenance algorithms, and three distance metrics. Each evolutionary run yielded one (best) solution, and runs for each combination of experimental settings were independently repeated 10 times. Therefore, the total number of solutions obtained for further analyses was $2 \cdot (1 + 5 \cdot 3) \cdot 10 = 320$.

6.3.2 Results

Diversity

Fig. 6.1 illustrates the distance between the best evolved solutions calculated using the `opt` measure and projected to 2D with multidimensional scaling (MDS) [44]. The structures evolved for the *height passive* goal exhibited more diversity than those evolved for the *velocity on land* goal. After the 2D projection, the solutions for the latter goal were more dispersed in both dimensions. Moreover, the percentage of preserved variance was much higher for the *velocity on land* goal (82%) than for the *height passive* goal (57%). For the former goal, the solutions obtained without any diversity maintenance technique formed a small cluster, while for the latter goal, they were more scattered. This result was expected, as the *height passive* goal involved larger structures and, as shown in Chapter 4, yielded multiple local optima that were morphologically distant according to the `opt` measure. For the *velocity on land* goal, the structures were smaller and less diverse in terms of morphology. However, they could differ with regard to their neural networks. Possibly, for tasks that involve behaviors (such as maximizing velocity), a distance measure that considers the lifetime performance, such as the movement properties [121], might be more beneficial than a measure that estimates the similarity of static body structures and shapes.

For the *velocity on land* task, where the solutions obtained using the standard EA were less diversified, the differences between diversity maintenance techniques and dissimilarity measures were more pronounced. Global niching resulted in the least diversification. Local

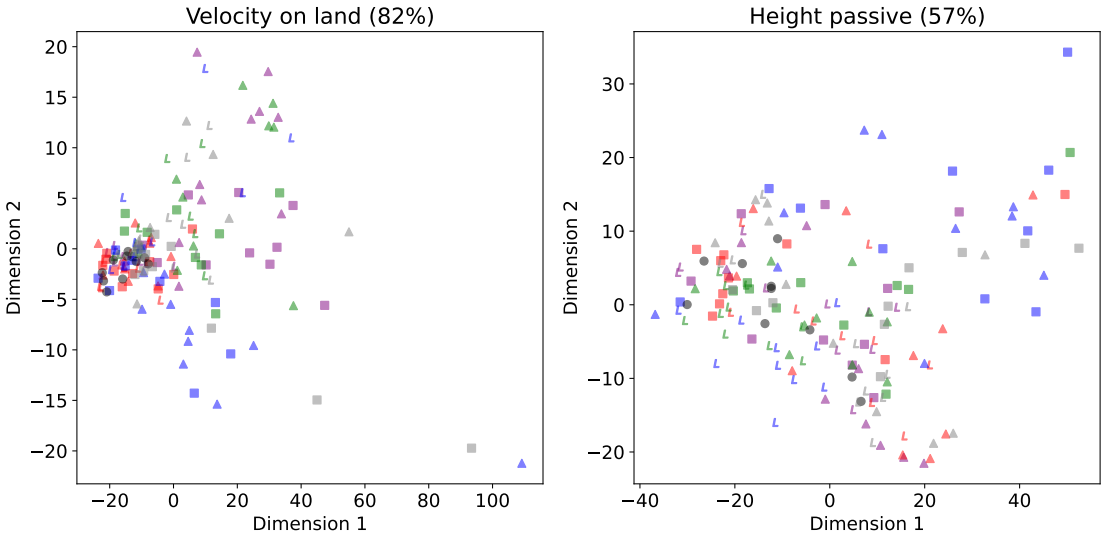


Figure 6.1: Distance between the best evolved solutions according to the `opt` measure, projected to 2D (fraction of variance preserved is shown in parentheses). Each plot shows $(1+5\cdot3)\cdot10$ solutions for one fitness criterion: *velocity on land* (left column) or *height passive* (right column). Diversity control: none (black •), niching global (red), niching local (purple), novelty global (blue), novelty local (green), NSGA-II (gray). Distance measures: gene (letter L), `opt` (▲), `shape` (■). In order to make overlapping symbols more readable, a random jitter was added to the location of each symbol (uniformly distributed, 3% of the range of each axis).

niching, which explored simultaneously more niches, produced more diversified structures, however, only when using the phenetic `opt` and `shape` measures. Novelty search, which had a strong exploration pressure, led to high diversity for both global and local versions and all three measures. Interestingly, for the *height passive* task, both niching and novelty yielded similarly diverse structures. This result may be attributed to the fact that the solutions in this task are in general more phenetically diverse. It is also worth noticing that for this task, the Levenshtein genetic distance measure produced similar levels of diversity as the phenetic distance measures.

Figs. 6.2 and 6.3 show individuals with the highest position of the center of mass evolved using the *f1* genetic encoding. The former figure presents the results of global niching with the `opt` distance measure, and the latter figure displays the results of global novelty search with the same measure. The figures indicate that while niching achieved higher fitness, both approaches facilitated the evolution of highly diverse structures.

Best fitness

Fig. 6.4 illustrates the fitness of the best individual in each generation for the *height passive* goal, obtained by the standard EA and five dissimilarity maintenance algorithms using the `opt` measure. The standard EA achieved the fastest convergence, mostly around the 500-th generation. NSGA-II and global niching exhibited a more gradual fitness improvement, resulting in higher fitness values, although NSGA-II also attained fast convergence in some runs. Local niching, which rewarded solutions for being locally diverse, had a lower overall fitness and lacked convergence. This trend was even more pronounced for novelty search,

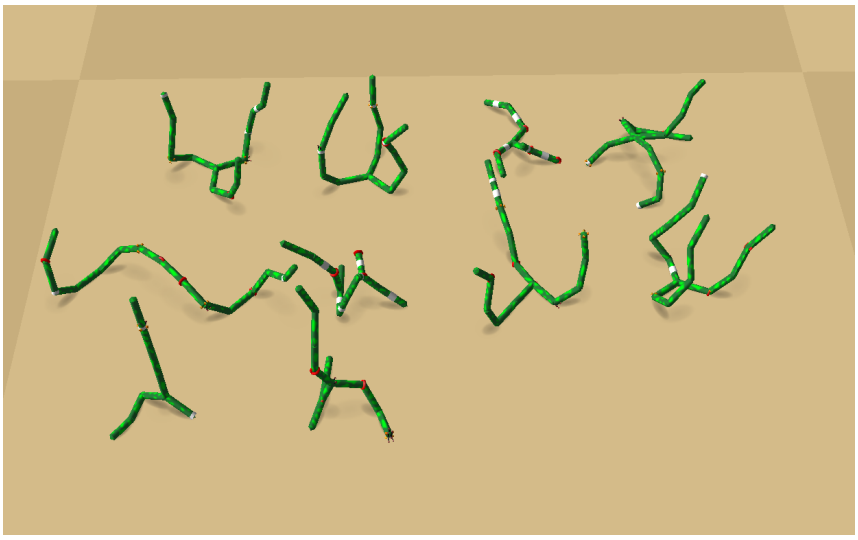


Figure 6.2: Best individuals evolved with the *f1* genetic encoding and global niching using the opt measure. Fitness of these structures is shown in the bottom plot in Fig. 6.5, "Niching Global opt" boxplot.

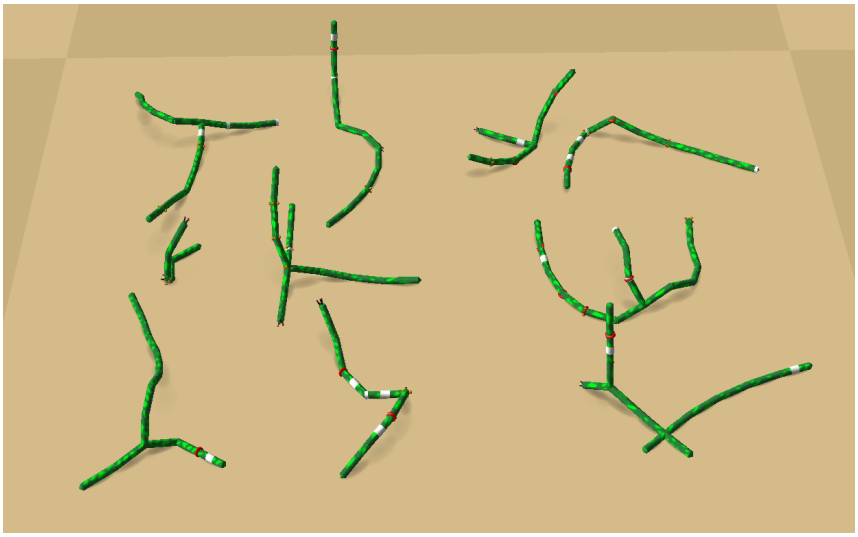


Figure 6.3: Best individuals evolved with the *f1* genetic encoding and global novelty search using the opt measure. Fitness of these structures is shown in the bottom plot in Fig. 6.5, "Novelty Global opt" boxplot.

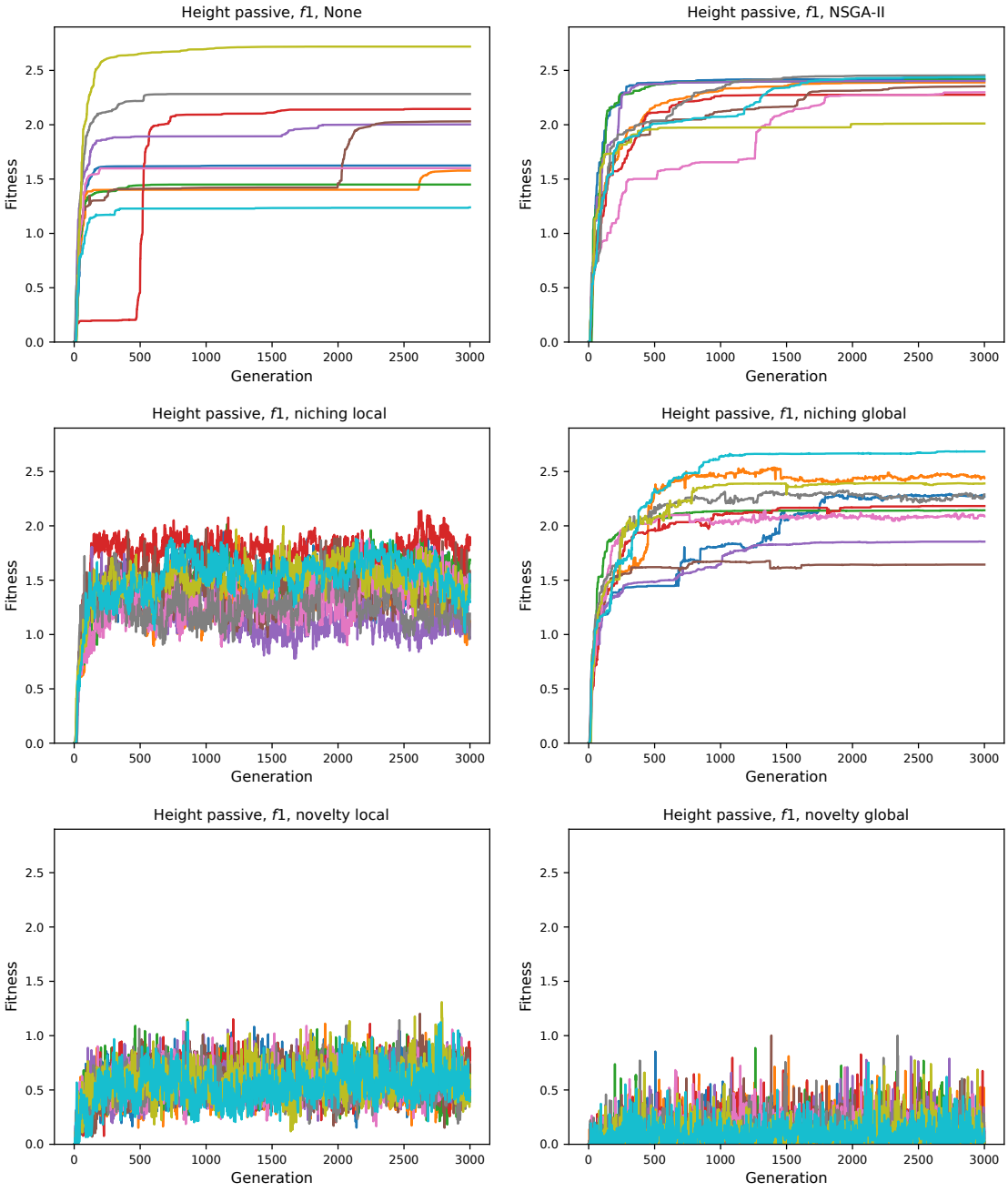


Figure 6.4: Best individual's raw fitness in each generation in the *height passive* task for standard EA (None), NSGA-II, local niching, global niching, local novelty, and global novelty, each using the opt distance measure. Colors correspond to different runs of the algorithm.

with global novelty having the lowest fitness values. It is worth noting that in this task, characterized by multiple phenetically distant local optima according to the `opt` measure, the local version of niching using this measure found more fit solutions than its global version.

The raw fitness distribution of the best evolved individuals across all runs is depicted in Fig. 6.5. As expected, novelty search yielded the lowest fitness values, since fitness was not considered during selection. The highest fitness values were achieved by NSGA-II with the `gene` measure. The effectiveness of the NSGA-II may be attributed to the fact that it optimizes both the objectives: fitness and novelty without aggregating them. A possible explanation for the good performance of the `gene` measure may be that it has the highest possible resolution because any difference in structures must be somehow reflected in their genotypes. However, the `gene` measure did not outperform the other measures when applied in other maintenance techniques.

The niching algorithms outperformed novelty search, but were inferior to NSGA-II, in terms of the best fitness. For the *velocity on land* task, local niching with the `shape` measure achieved the highest median. For the *height passive* task, global niching with the `opt` measure produced the highest median. These results suggest that the effectiveness of the diversity maintenance techniques in terms of fitness depended not only on the degree of the diversity pressure, appropriate to the search space characteristics, but also on the proper dissimilarity measure that reflect the relevant properties of the solutions. This is further supported by the fact that NSGA-II, the best-performing algorithm overall, achieved lower results in terms of the best fitness median than the niching when using the `shape` measure in the *height passive* task.

Computational cost

A single evolutionary run took from on average 7.5 to 211 hours on a single core of an Intel Xeon Gold 5320 2.20GHz CPU. The experiments using the `shape` measure were the most computationally intensive – they lasted from on average 88 to 211 hours. For this measure, the number of samples affects the complexity the most and it can be traded off for accuracy or computation time. The experiments using the `opt` measure were less computationally expensive, ranging from on average 30.5 to 77 hours per run. The standard EA run took on average 7.5 hours.

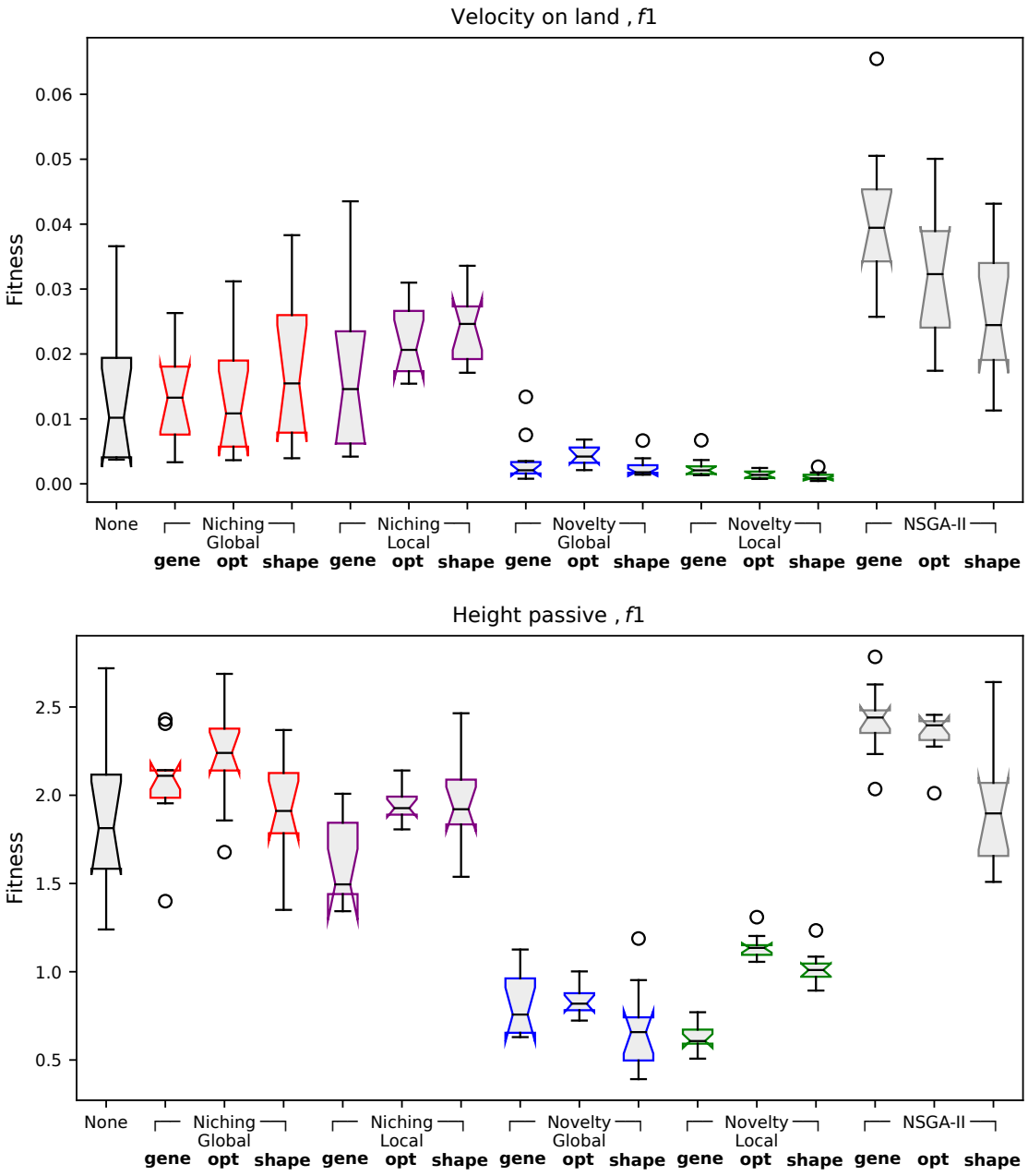


Figure 6.5: Fitness of the best evolved solutions. Upper plot: *velocity on land*, bottom plot: *height passive*. Each plot shows 16·10 solutions. Diversity control: *None*, *Niching Global*, *Niching Local*, *Novelty Global*, *Novelty Local*, *NSGA-II*. Distance measures: gene, `opt`, shape.

6.4 Summary

This chapter investigated how the selection of the diversity maintenance algorithm and the distance measure affects the evolution of 3D structures in terms of the best final fitness and the diversity of obtained solutions. Two optimization tasks were considered, one active and one passive. All three distance measures (`gene`, `opt`, and `shape`) used by the diversity promotion techniques increased the diversity of solutions, but their performance varied depending on the optimization task and the diversity maintenance algorithm. In the more challenging *velocity on land* task, some combinations of the algorithms and measures resulted in less diverse solutions than others.

Both of the analyzed methods of encouraging diversity – novelty search and niching – were generally equally effective in terms of the diversity of solutions, especially in the *height passive* task. However, the solutions found by novelty search had lower fitness values. On the other hand, niching produced solutions that were both diverse and had higher fitness values than those obtained using standard EA. Moreover, niching using the `opt` measure achieved fitness comparable to the NSGA-II algorithm in the *height passive* task. These results indicate that niching facilitated broader exploration of the solution space, which led to the discovery of better local optima.

Interestingly, the use of a simple genetic-based distance measure for novelty assessment in NSGA-II produced the highest-fit individuals. The Levenshtein distance metric is simple, fast, and mostly problem-independent, unlike phenetic distance measures that require a lot of time to develop and test. This implies that genetic distance measures or low-level representation-based measures can be advantageous for any optimization problem and should be considered as a part of the metrics set for promoting diversification.

The results indicated that evolving 3D structures for height and velocity without using the fitness function do not produce highly-fit solutions. When the fitness function is used to guide the evolutionary search and a diversity maintenance technique is applied, it is advantageous to employ a dissimilarity measure that reflects the relevant properties of the solutions.

7 Summary

7.1 Goals

The aim of this thesis was to explore the use of dissimilarity measures to enhance the effectiveness of the evolutionary algorithms in the challenging problem of evolutionary design (ED) of 3D structures. For this purpose a data set of active and passive 3D structures and a set of diversified dissimilarity measures was required. After the development of the data set and the dissimilarity measures, the first objective was to investigate global convexity of fitness landscapes – a property that has been in the past exploited in combinatorial optimization problems – using the fitness-distance correlation analysis. The next objective was to develop mutation and crossover operators that aimed to control the distance between parent and child solutions. Finally, the last goal was to compare the performance of different maintenance algorithms and dissimilarity measures in the diversity maintenance techniques.

7.2 Contributions

The author's key contributions are summarized below:

- The development of a data set of passive and active 3D structures (Chapter 2). The data set consists of structures obtained in a series of evolutionary experiments optimizing four objectives: velocity on land, velocity in water, height of active structures, and height of passive structures. The data set covers a wide range of structures that vary in their morphology, neural complexity, and fitness. Therefore, it is well-suited for exploring the characteristics of the four aforementioned ED problems.
- The development/adaptation of dissimilarity measures for active and passive 3D structures (Chapter 3). In the experiments two existing dissimilarity measures were used: `gene` – the Levenshtein distance for genotypic dissimilarity and `greedy` – a heuristic measure based on comparing graphs representing the structures. Additionally, three measures were developed as a part of this work: `shape` – a measure based on shape descriptors, and two measures based on the spatial distribution of structures: `dens` and `freq`. Each of the measures can be used in any application where a 3D structure can be represented as an undirected graph (or a solid shape

model in the case of `shape`, `dens`, and `freq` measures). The `opt` measure also allows incorporating additional vertex properties into the dissimilarity calculation.

- The investigation of human perception of the similarity between 3D structures (Chapter 3). The study introduced and fine-tuned the methodology of investigating human perception of similarity of 3D structures represented as undirected graphs and provided the preliminary results.
- The analysis of the global convexity of the selected ED problems (Chapter 4). In the study, the fitness-distance correlation (FDC) was calculated in four optimization tasks using three genetic representations and five dissimilarity measures, one genetic and four phenetic.
- The development and analysis of the enhanced mutation and crossover genetic operators employing dissimilarity measures for 3D structures to improve the efficiency of the evolutionary search (Chapter 5).
- The analysis of the effect of diversity maintenance mechanism and distance measure on the evolution of 3D structures in the selected ED problems (Chapter 6).

The investigation of the human perception of similarity of 3D structures showed that humans tend to assess similarity according to the perceived functional correspondence between the structures. This result may suggest that a dissimilarity measure based on human judgments of similarity could potentially be beneficial in ED tasks. On the other hand, the functional similarity perceived by humans may not reflect the true functional similarity (e.g. a part of a structure perceived as a "tail" may actually serve as a "leg"). Therefore, further research is needed to verify this hypothesis.

The fitness-distance correlation analysis showed that FDC values vary depending on the genetic representation, the optimization task, and the employed dissimilarity measure, indicating, in some of the cases, the global convexity of fitness landscape. This result suggests that for the genetic encodings, optimization goals, and dissimilarity measures for which highly negative FDC values were obtained it may be beneficial to apply distance-preserving genetic operators employing those measures.

The proposed TSM operator, aiming to control the dissimilarity between the parent and offspring solution, was shown to outperform the native mutation operator for most of the cases. The newly developed dissimilarity-based crossover operators yielded results comparable to the native operators in terms of the best fitness. The results suggests that to increase the performance of crossover it may be necessary to develop distance-preserving operators not relying on the native ones.

The analysis of the performance of diversity maintenance algorithms combined with different dissimilarity measures indicated that both the degree of diversity pressure and the selected distance measure influence the results in terms of fitness for a given problem. It was also demonstrated for considered optimization objectives that niching and novelty search produced, in general, similarly diversified structures. The simple genotype-based measure was found to be effective in producing diverse and high-fitness solutions, especially when used in the NSGA-II algorithm.

7.3 Future work

The work presented in this thesis suggests directions for future research, the main of which are summarized below:

- Developing a dissimilarity measure that combines morphological and performance aspects (such as the movement properties) to be applied in the genetic operators and diversity maintenance techniques.
- Conducting a more comprehensive investigation of the human perception of similarity and developing a dissimilarity measure based on the features of 3D models that correlate with human assessment of similarity.
- Developing a dissimilarity crossover operator that will not rely on the native operators.
- Combining the diversity maintenance techniques with the enhanced genetic operators using different dissimilarity measures to further increase the effectiveness of the search.
- Investigation of more quality-diversity algorithms, such as NSLC, combined with different dissimilarity measures.

The present results and future directions show promise for enhancing the effectiveness of evolutionary design and for taking a step forward to tap its full potential.

Appendix A

Applications of evolutionary design

Year	Problem	Reference
1995	Control System Optimization	[211]
1995	Design of truss structures	[177]
1995	Composite laminate design	[137]
1995	Molecules design	[218]
1996	Wind turbine design	[193]
1996	Design of stiffened composite panels	[159]
1997	Desgin of electronic circuits	[156]
1997	Antenna design	[102]
1997	Design optimization of trusses	[178]
1997	Design of reinforced concrete beams	[40]
1997	Design of 3D modular manipulators	[38]
1998	Design of induction motors	[225]
1998	Design of IIR filter	[209]
1998	Design optimization of reinforced concrete frames	[179]
1998	Design optimization of aircraft wing planform	[161]
1999	Design of electronic devices and circuits	[205]
1999	Molecules design	[68]
2000	Design of combinatorial logic circuits	[39]
2000	Wing-shape optimization	[162]
2000	Fashion design	[111]
2001	Composite laminate design optimization	[199]
2001	Design of truss structures	[46]
2001	Design of robot's morphology and control system	[168]
2002	Robot design	[195]
2002	Antenna design	[151]
2002	Mechanical design components	[66]
2002	Circuit design	[36]
2003	Design of VLSI circuits	[41]
2003	Multi-speed gearbox design	[47]
2003	Neural network design	[32]
2004	Design of composite channels	[97]
2004	Evolving robot behaviour	[233]
2005	Underground excavation shape optimization	[183]
2005	Design of steel structures in tall buildings	[110]
2005	Evolving robot behavior	[208]
2006	Antenna design	[83]
2006	Antenna design	[64]
2006	Jewelry design	[223]
2006	Truss optimization	[212]

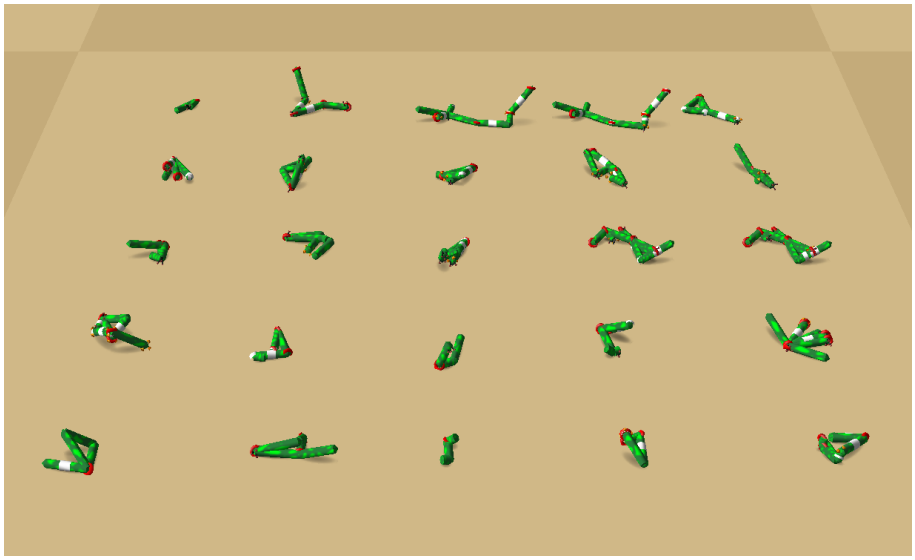
Year	Problem	Reference
2007	Analog circuit design	[160]
2007	Design of IIR filter	[227]
2007	Design of rolling element bearings	[180]
2007	Antenna design	[79]
2008	Design of periodic structures	[90]
2008	Design of modular robotic arms	[37]
2009	Design of Broadband Hybrid Coupler	[229]
2009	Architectural layout design	[226]
2009	Optimization of robot grippers	[189]
2009	Design of shell-and-tube heat exchangers	[173]
2009	Shape design of industrial electromagnetic devices	[51]
2009	Design of water tanks	[12]
2009	Desgin of medical parallel robots	[202]
2009	Robot design	[185]
2010	Design of digital IIR filter	[222]
2010	Turbo pump, compressor, and micro-air vehicles	[145]
2010	Building design optimization	[214]
2010	Shape design of Y-noise barriers	[73]
2010	Robot design	[23]
2011	Design of plate-fin heat exchangers	[10]
2011	Topology optimization of composite structures	[206]
2011	Airfoil shape optimization	[96]
2011	Optimization of manipulator arm morphology	[50]
2011	Aerodynamic design	[2]
2012	Radial basis function neural network design	[176]
2012	Truss optimization	[170]
2012	Nozzle design optimization	[163]
2012	Platformer design	[42]
2012	Antenna array design	[33]
2012	Antenna design	[31]
2012	Pole shape optimization	[10]
2012	Design of multilayered composite structure	[138]
2013	Design of condenser	[221]
2013	Antenna design optimization	[149]
2013	Aerodynamic design	[93]
2014	Shape optimization	[213]
2014	Design of counterrotating compressors	[101]
2014	Design of lattice opto-materials	[91]
2014	Airfoil shape optimization	[49]

Year	Problem	Reference
2014	Analog circuit design optimization	[13]
2014	Fixture design, fixture analysis, fixture synthesis, fixture layout design, optimization of fixture layout design	[217]
2015	Design of lattice structure	[210]
2015	Preform design for forging of 3D blade	[194]
2015	Shape optimization of free-form steel space-frame roof structures	[115]
2015	PCB design	[21]
2015	Robot design	[43]
2016	Topological design for phononic band gap crystals	[144]
2016	Design of foam filled sandwich panels	[106]
2016	Structural design of stamping die components using bi-directional evolutionary structural optimization method	[6]
2016	Spatial architecture layout design	[53]
2016	Design of an exoskeleton for finger rehabilitation	[15]
2017	Design of straight bevel gears	[232]
2017	Spatial architecture layout design	[74]
2017	Antenna design	[72]
2017	Optimization of retaining wall design	[65]
2018	Airfoil shape optimization	[219]
2018	Design of cam-roller follower mechanism	[76]
2019	Design of shallow foundation	[107]
2019	Topology optimization of thin-walled square tubes	[9]
2019	Design of water distribution network	[157]
2019	Airfoil shape optimization	[146]
2019	Building design optimization	[63]
2019	Airfoil shape optimization	[146]
2020	Evolving neural networks	[153]
2020	Topology optimization	[220]
2020	Design of sewer networks	[78]
2021	Wind farm layout optimization	[113]
2022	Design of frame structures	[228]
2022	Topology optimization	[231]
2023	Design of PID controller	[100]

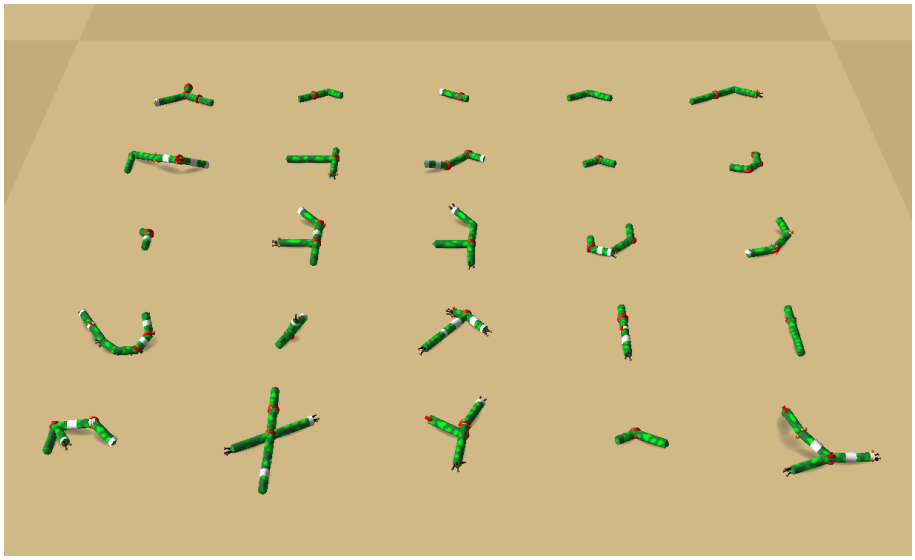
Table A.1: Works in the evolutionary design that did not meet the selection criteria of the literature review (Chapter 1).

Appendix B

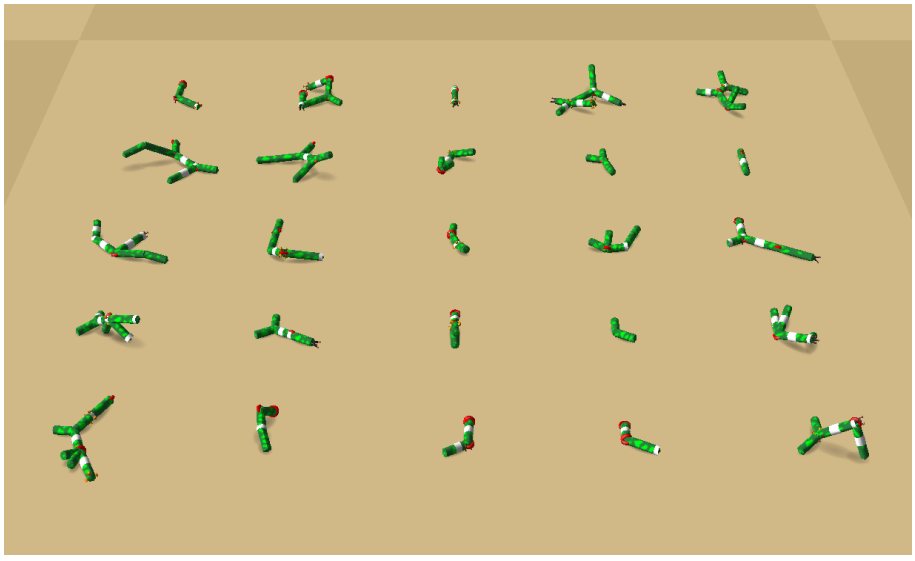
Structures from 1000_data_set



(a) *f0*

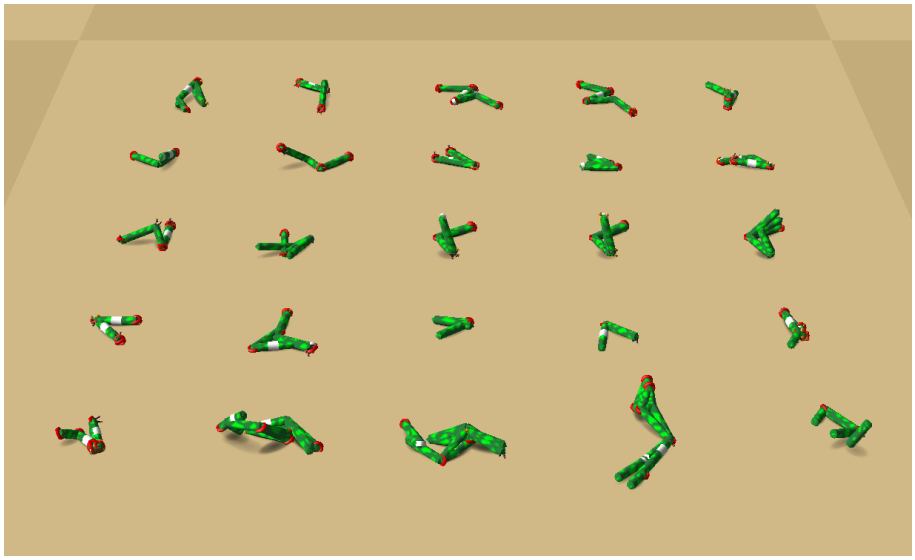


(b) *f1*

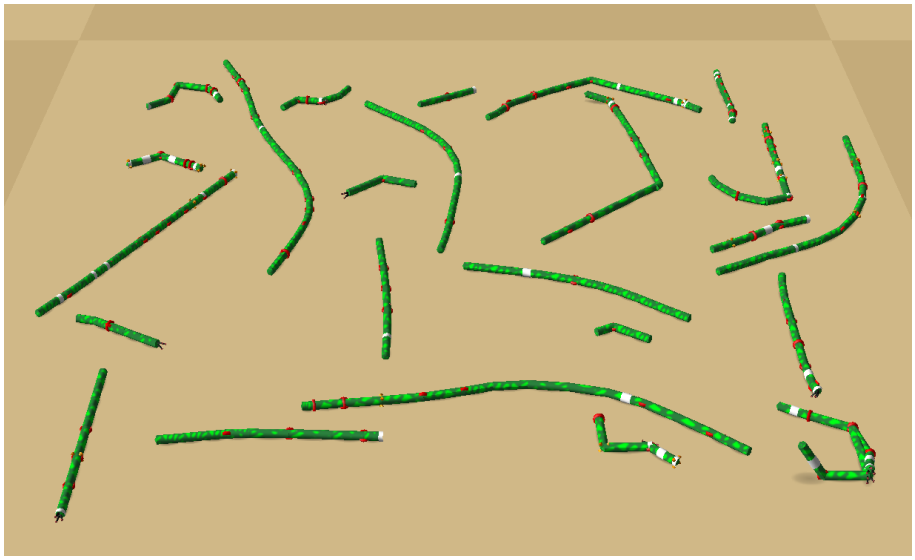


(c) *f4*

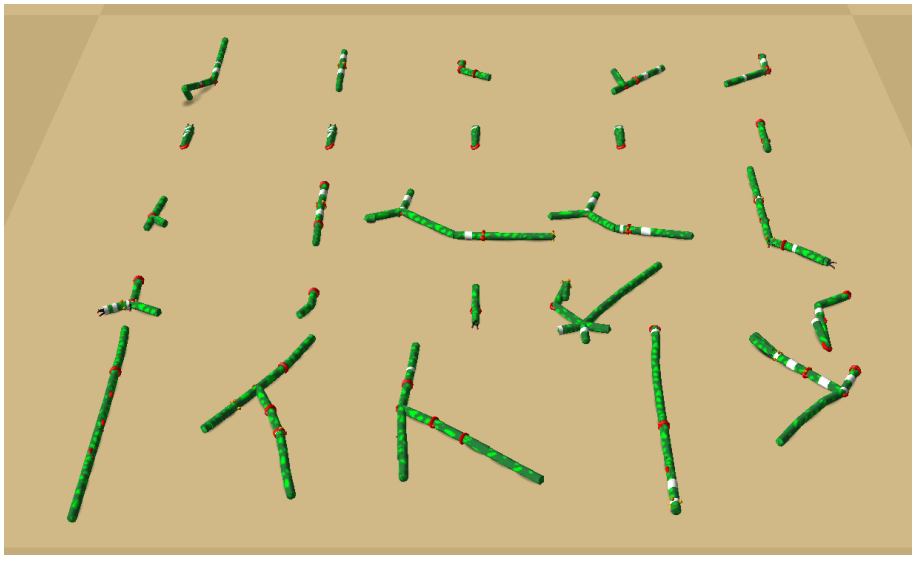
Figure B.1: A subset of 25 representative structures from the `1000_data_set` evolved for maximizing velocity on land.



(a) *f0*

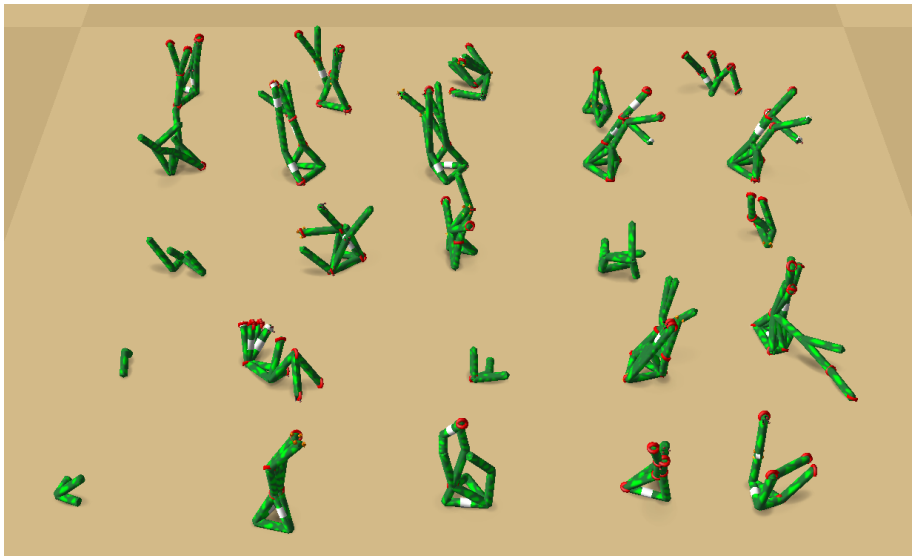


(b) *f1*

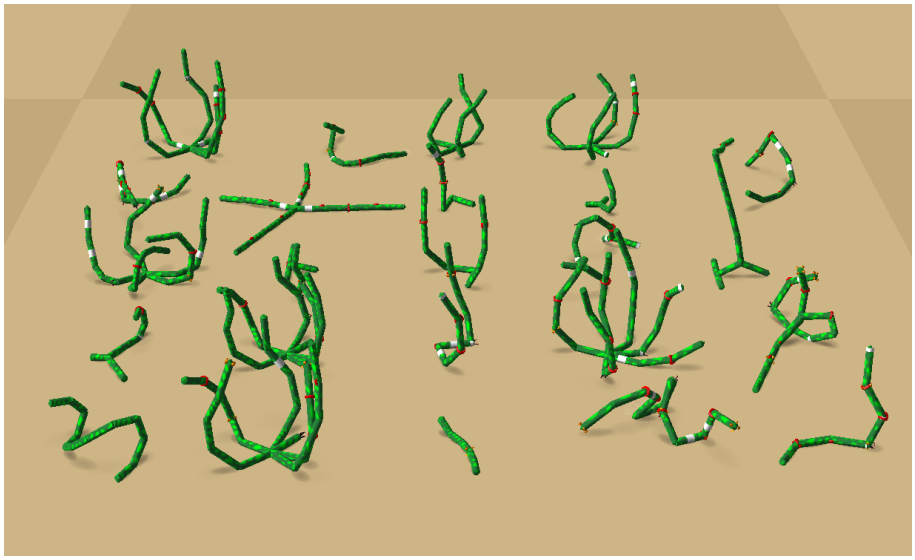


(c) *f4*

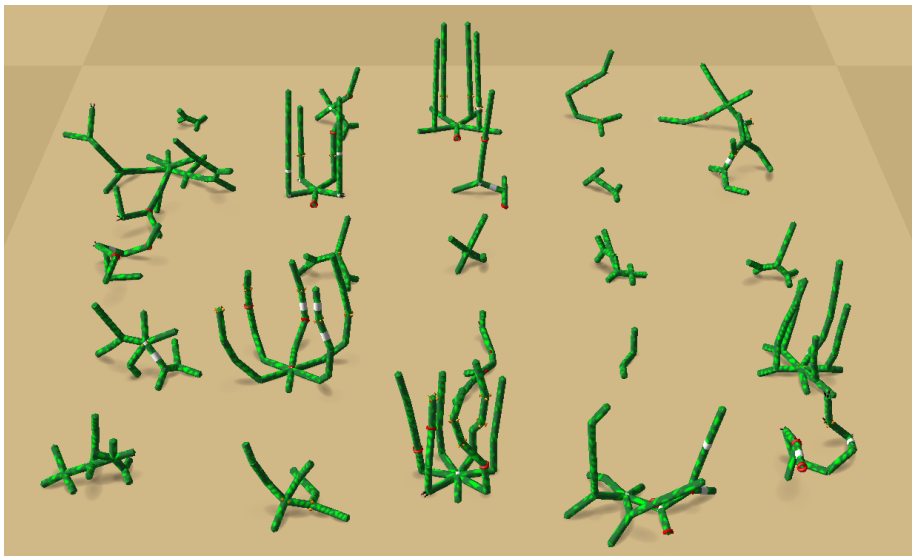
Figure B.2: A subset of 25 representative structures from the `1000_data_set` evolved for maximizing velocity in water.



(a) *f0*

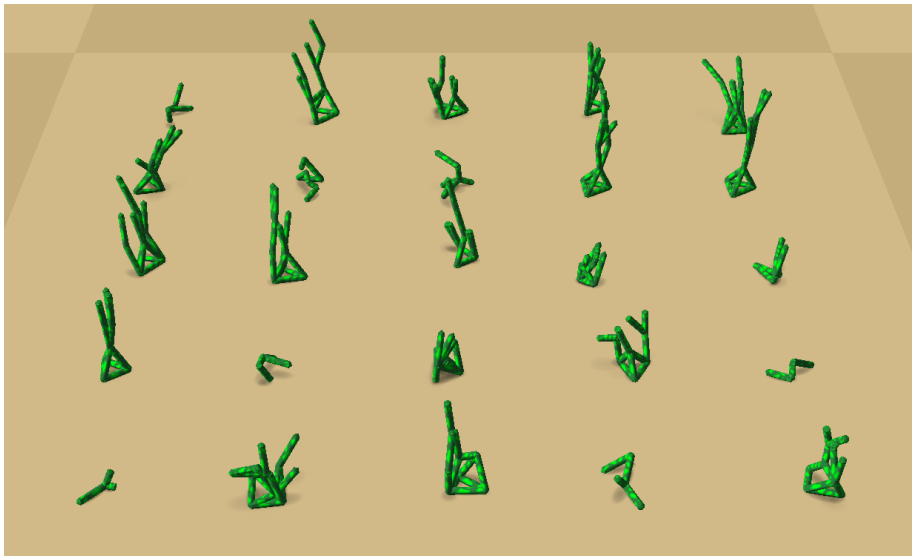


(b) *f1*

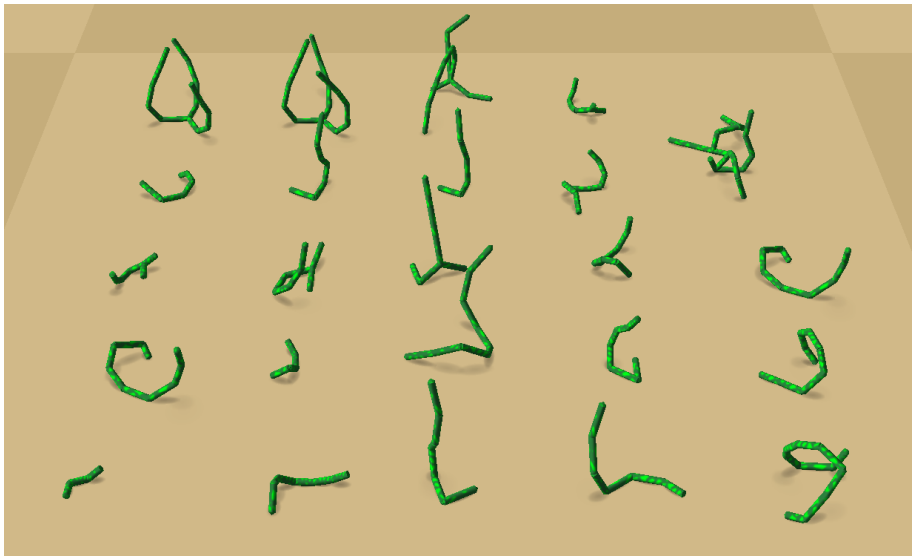


(c) *f4*

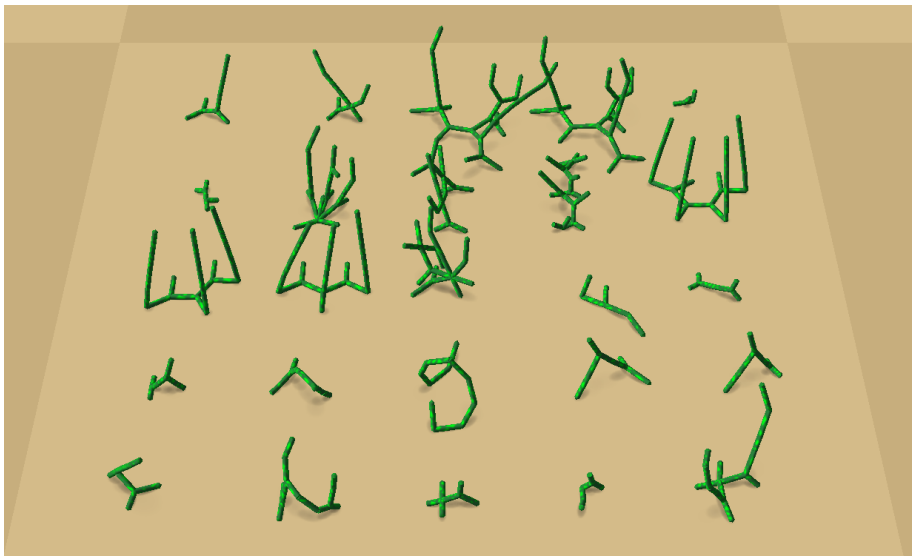
Figure B.3: A subset of 25 representative structures from the `1000_data_set` evolved for maximizing vertical position (active structures).



(a) *f0*



(b) *f1*



(c) *f4*

Figure B.4: A subset of 25 representative structures from the `1000_data_set` evolved for maximizing vertical position (passive structures).

Appendix C

Parameter tuning of the shape measure

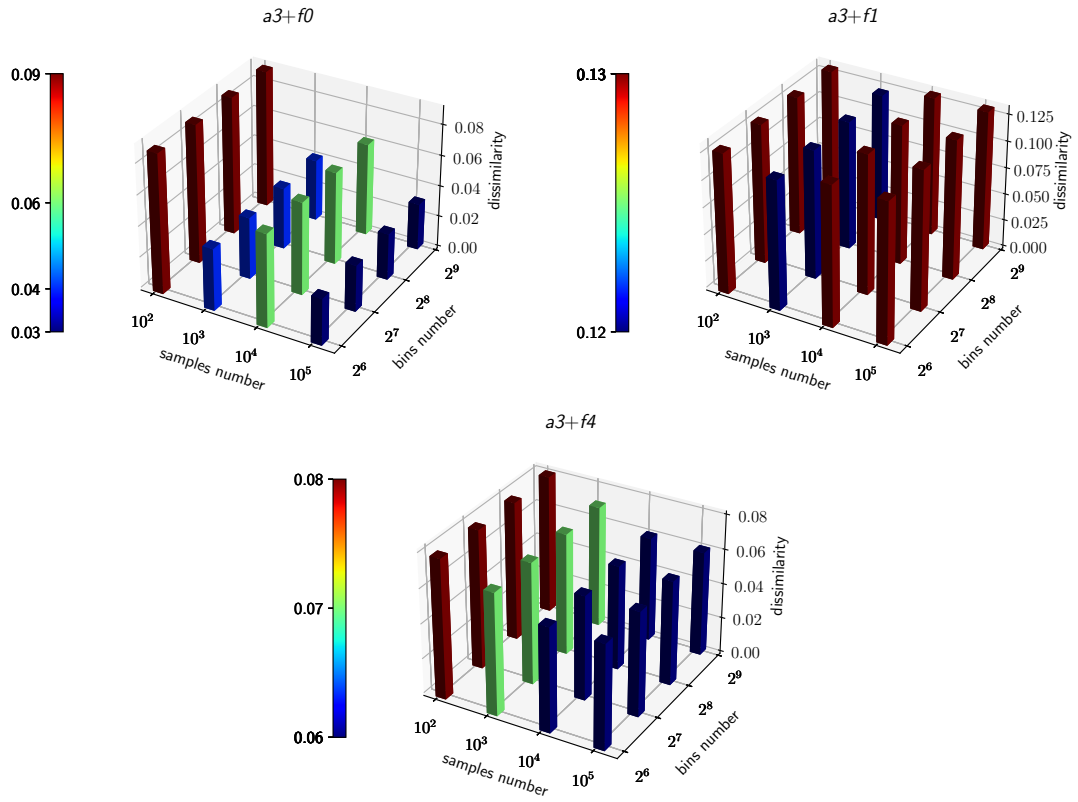


Figure C.1: Dissimilarity value for various combinations of `samples number` and `bins number` using descriptor a3. The color of each bar indicates the dissimilarity value.

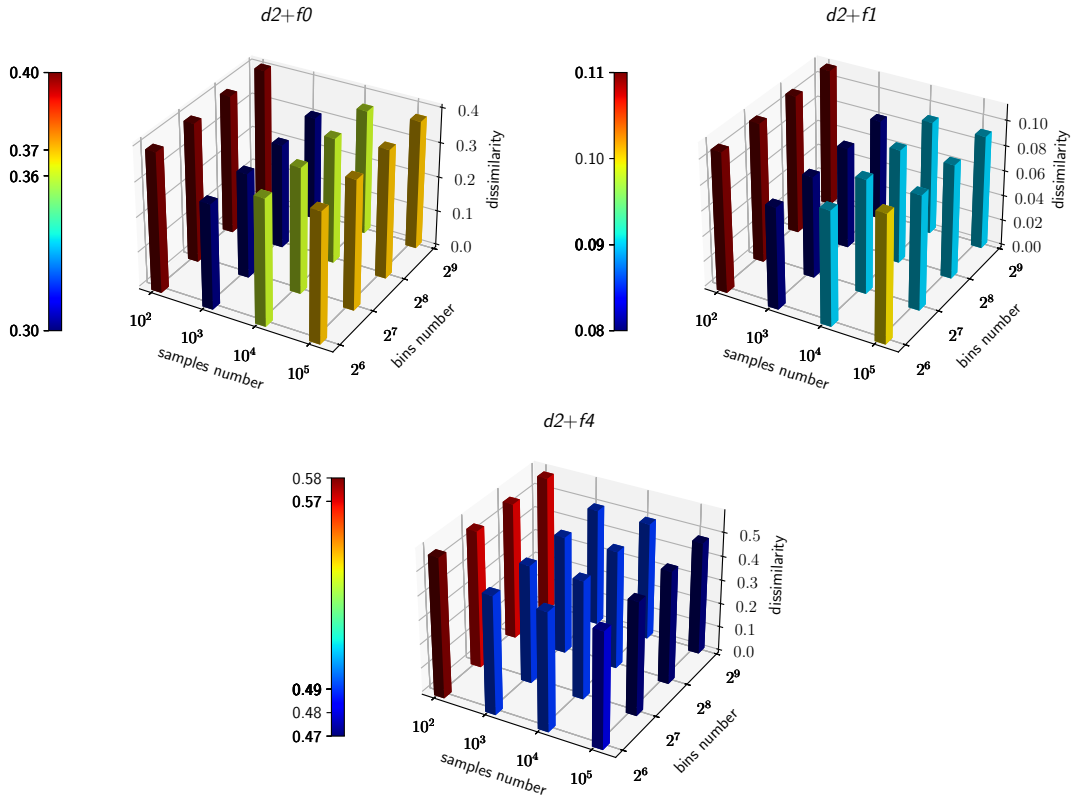


Figure C.2: Dissimilarity value for various combinations of `samples number` and `bins number` using descriptor `d2`. The color of each bar indicates the dissimilarity value.

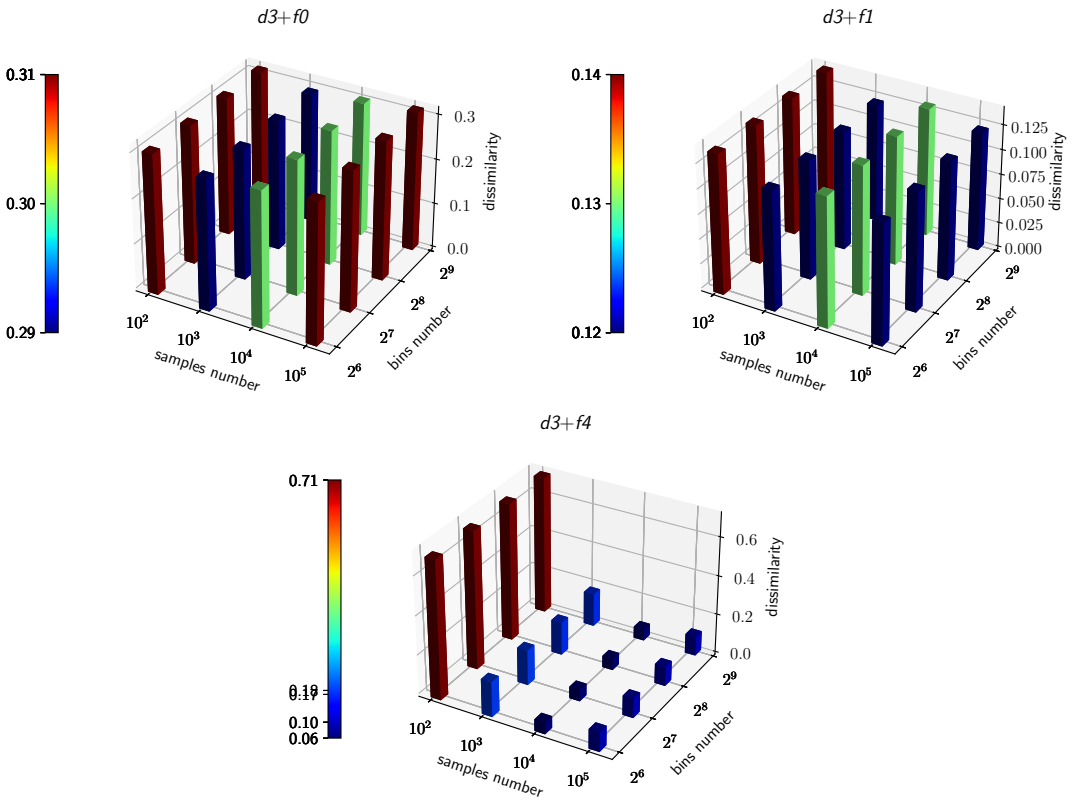


Figure C.3: Dissimilarity value for various combinations of `samples number` and `bins number` using descriptor `d3`. The color of each bar indicates the dissimilarity value.

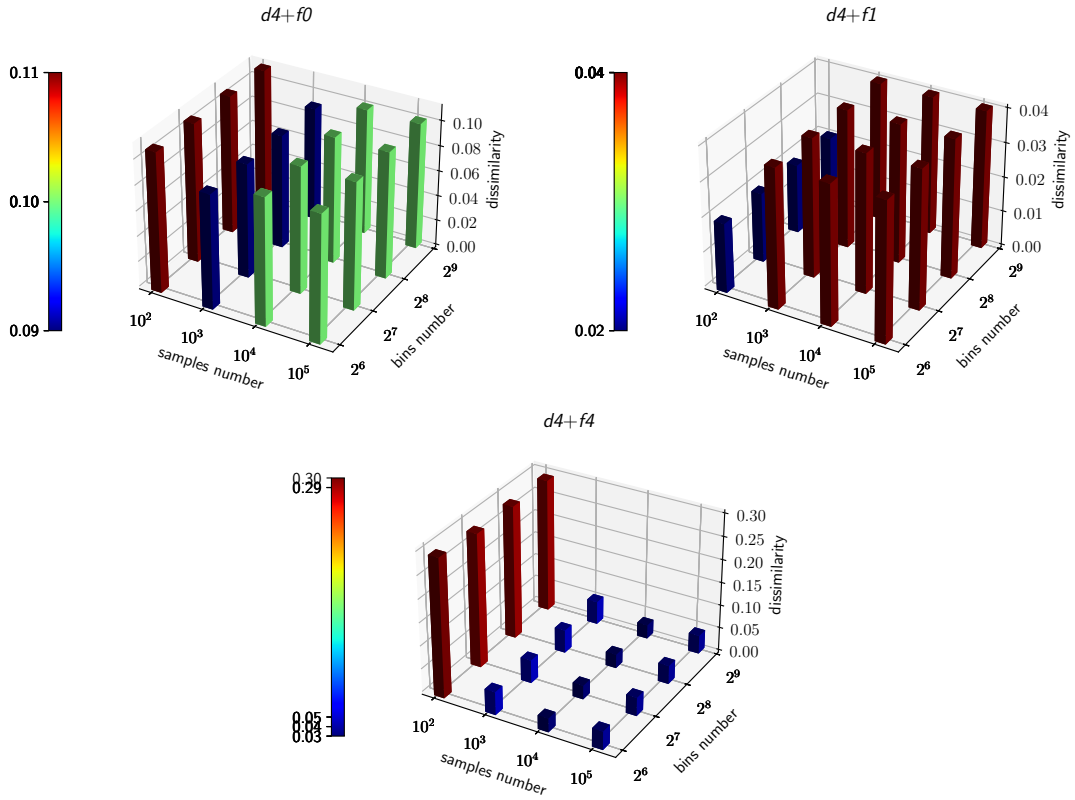


Figure C.4: Dissimilarity value for various combinations of `samples number` and `bins number` using descriptor `d4`. The color of each bar indicates the dissimilarity value.

Appendix D

Structures used for dissimilarity measures comparison

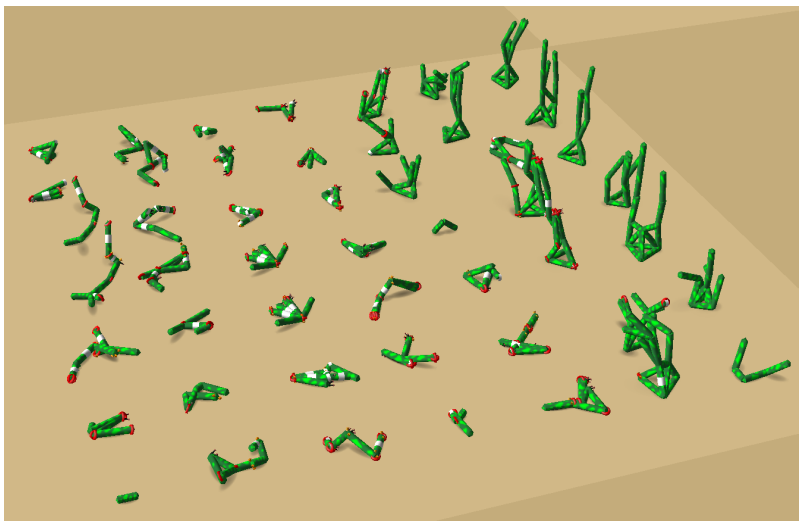


Figure D.1: A subset of 49 representative structures from the test set for the *f0* representation.

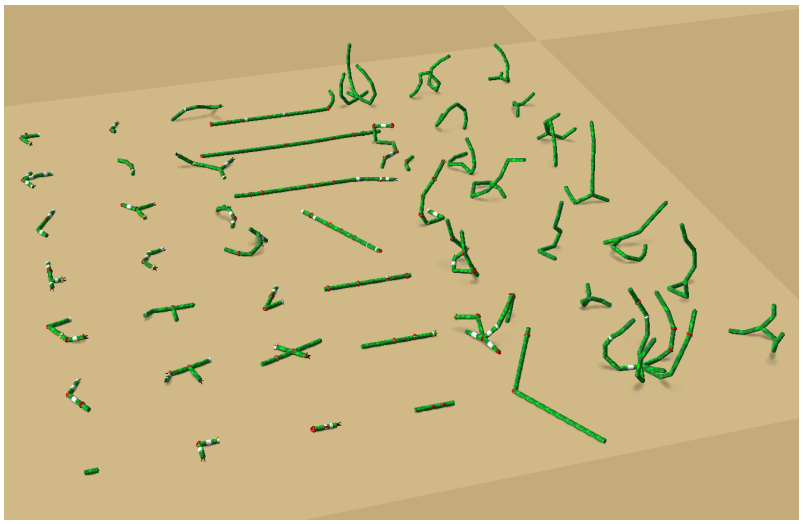


Figure D.2: A subset of 49 representative structures from the test set for the *f1* representation

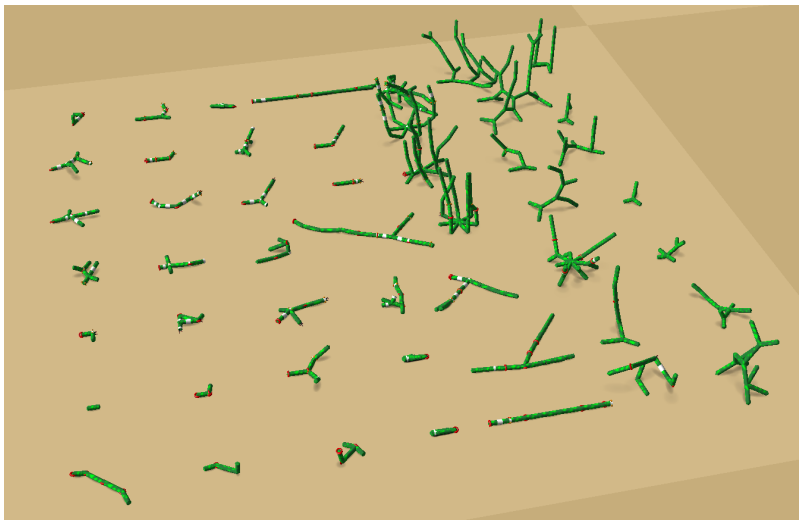


Figure D.3: A subset of 49 representative structures from the test set for the *f4* representation.

Appendix E

Correlations between the dissimilarity measures

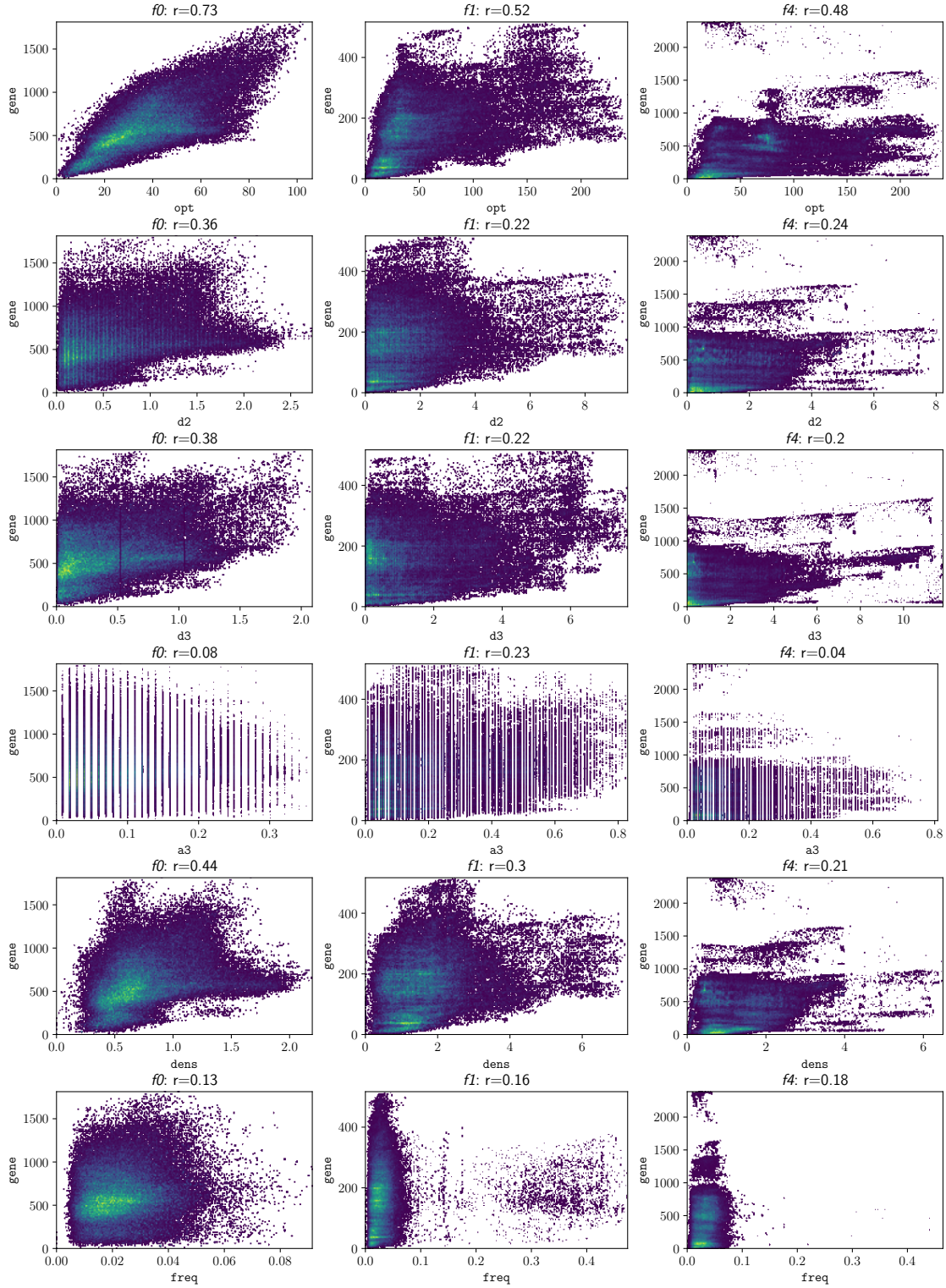


Figure E.1: Correlations between the gene measure and the other measures. Each point on the plot represents the dissimilarity value for a pair of structures encoded using a given genetic encoding. The color indicates the density of points, with lighter colors corresponding to higher densities.

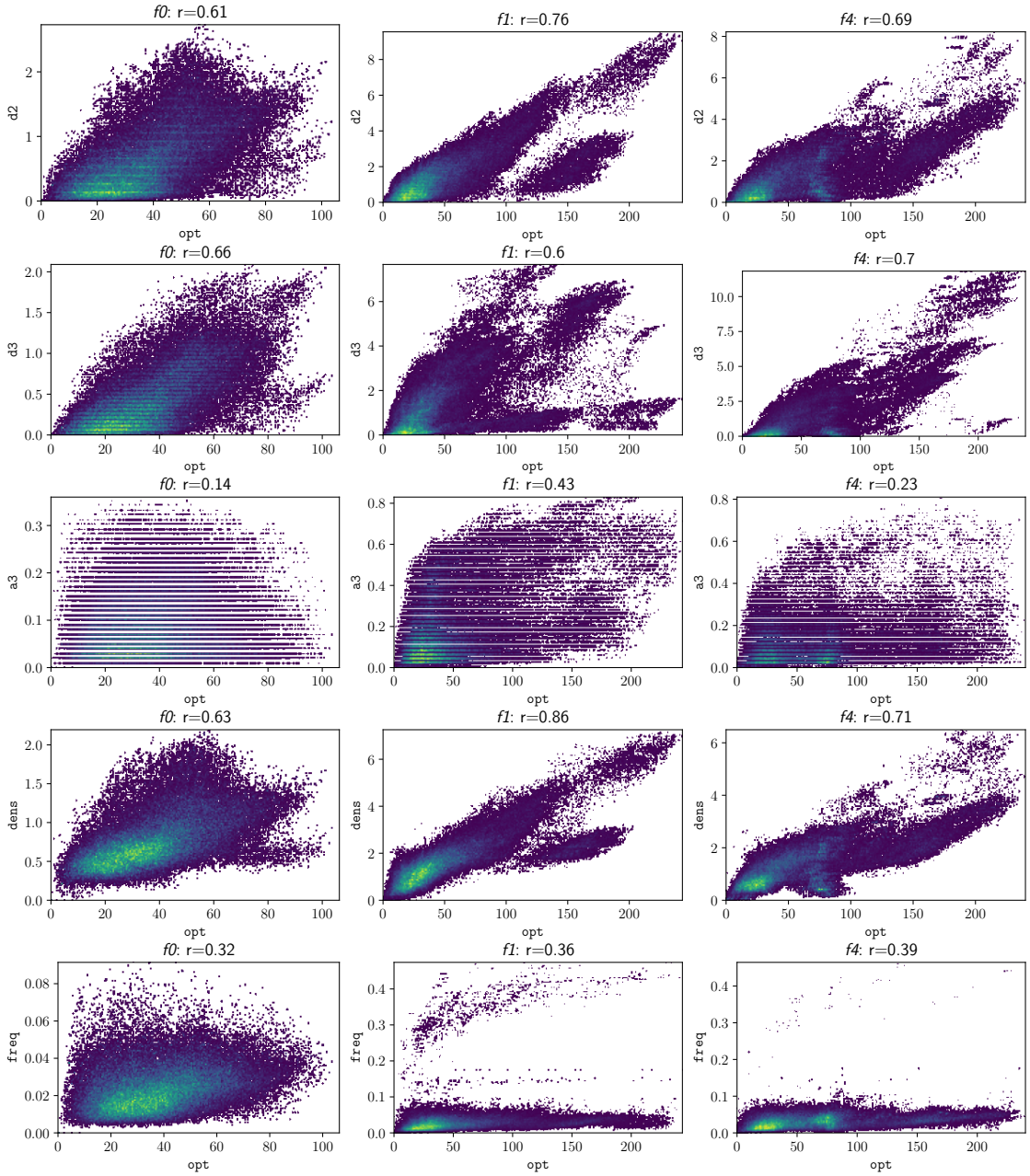


Figure E.2: Correlations between the `opt` measure and the other measures. Each point on the plot represents the dissimilarity value for a pair of structures encoded using a given genetic encoding. The color indicates the density of points, with lighter colors corresponding to higher densities.

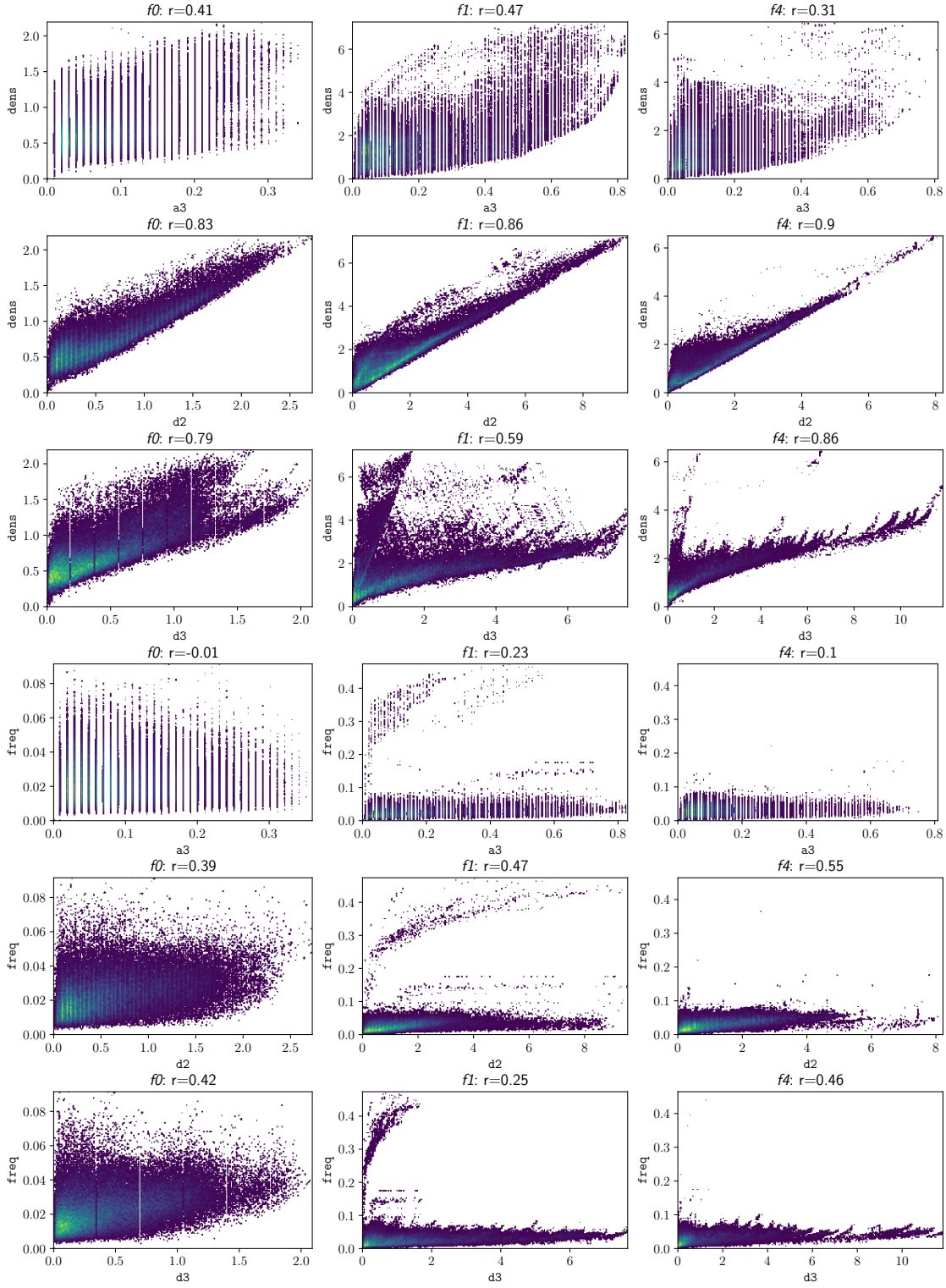


Figure E.3: Correlations between the shape measure and the other measures. Each point on the plot represents the dissimilarity value for a pair of structures encoded using a given genetic encoding. The color indicates the density of points, with lighter colors corresponding to higher densities.

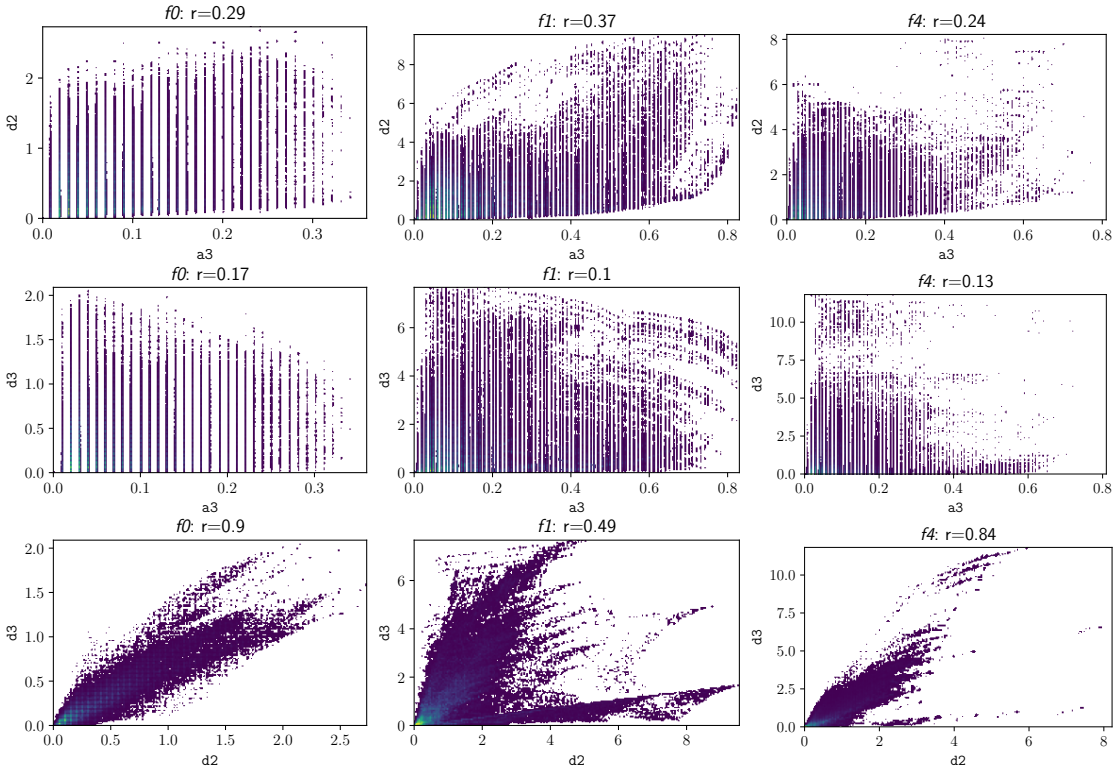


Figure E.4: Correlations between the shape measure calculated using descriptors a3, d2, d3. Each point on the plot represents the dissimilarity value for a pair of structures encoded using a given genetic encoding. The color indicates the density of points, with lighter colors corresponding to higher densities.

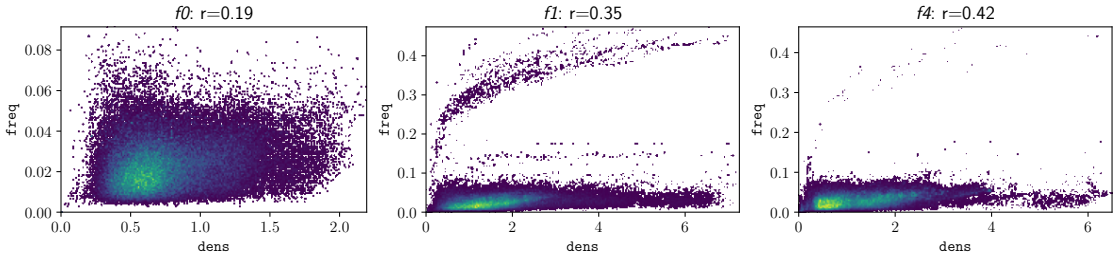


Figure E.5: Correlations between shape and `freq` measures. Each point on the plot represents the dissimilarity value for a pair of structures encoded using a given genetic encoding. The color indicates the density of points, with lighter colors corresponding to higher densities.

Appendix F

Human study participants' characteristics

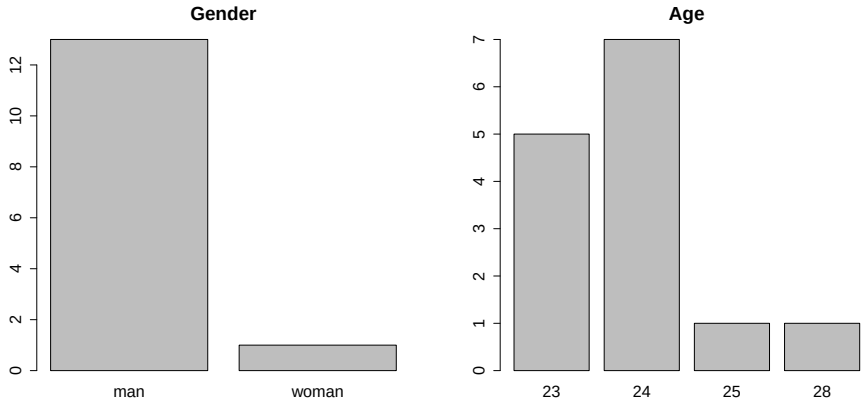


Figure F.1: The distribution of gender and age of the participants [120].

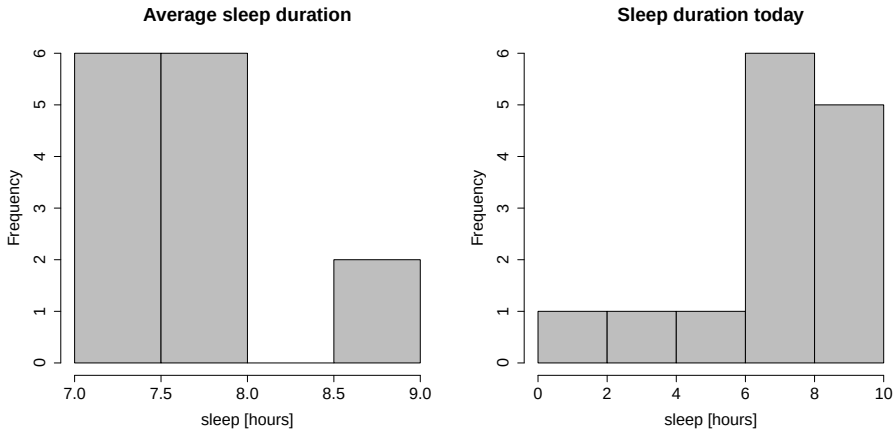


Figure F.2: Average sleep duration and sleep duration on the night preceding the investigation [120].

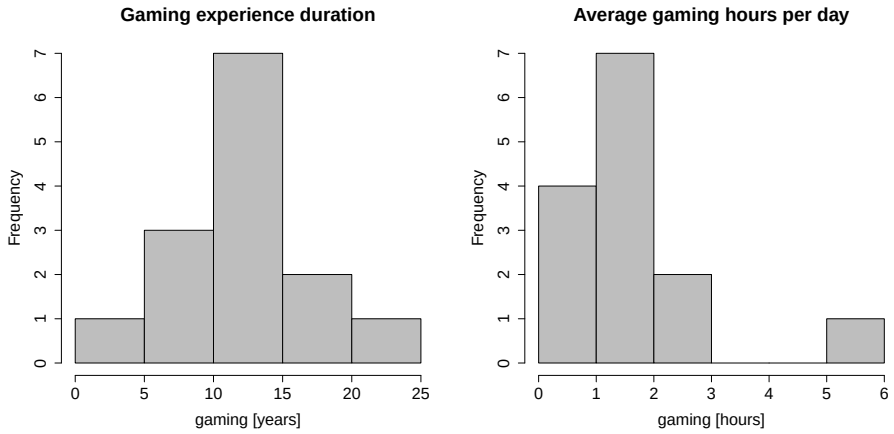


Figure F.3: Number of years of playing video games and the average number of playing video games per day [120].

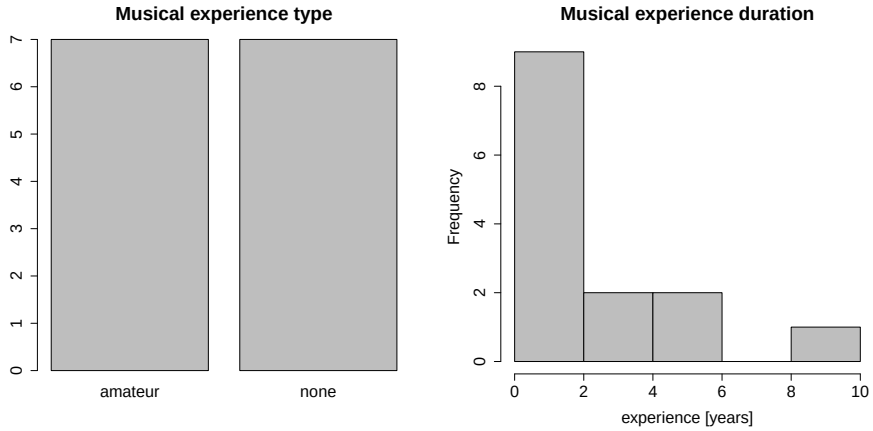


Figure F.4: Level of musical education of participants and their number of years devoted to musical education [120].

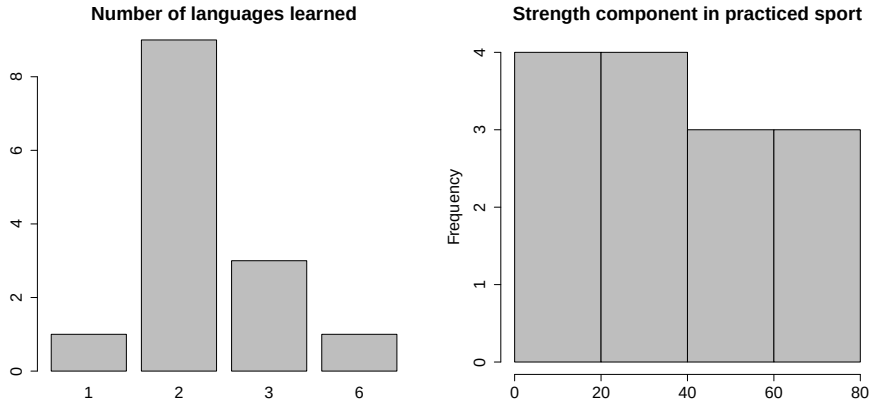


Figure F.5: Number of foreign languages known by participants and the percentage of strength component in their practiced sport [120].

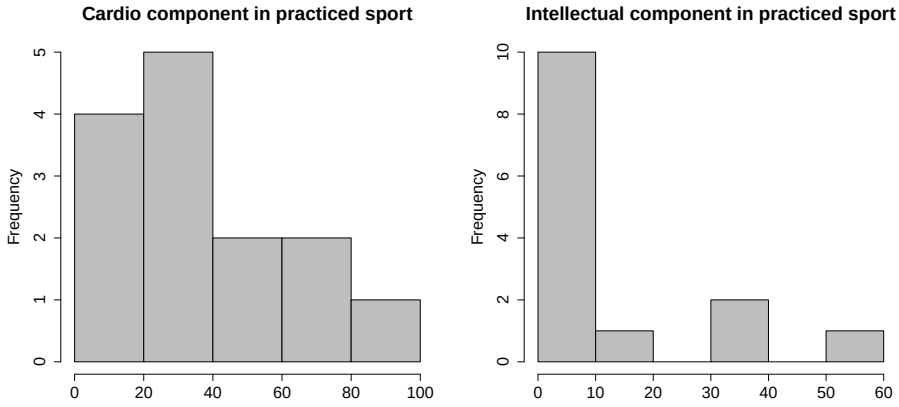


Figure F.6: The percentage of endurance and intellectual components in sports practiced by participants [120].

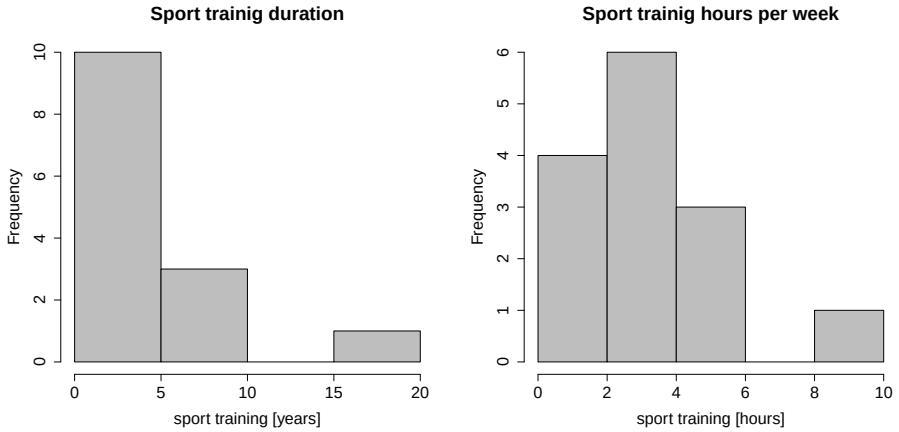


Figure F.7: Number of years of sport training and the average number of hours per week devoted by participants for sports training [120].

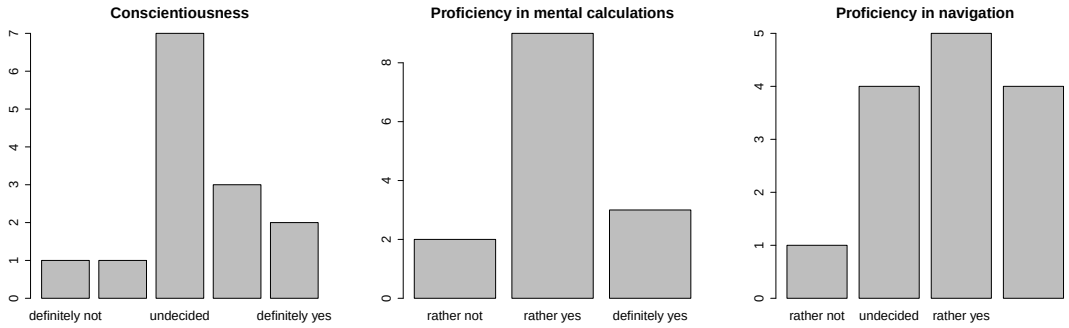


Figure F.8: The distribution of self-assessed conscientiousnesses, proficiency in calculations and navigation. The possible answers were: "definitely not", "rater not", "undecided", "rather yes", "definitely yes" [120].

Appendix G

Enhanced crossover operators

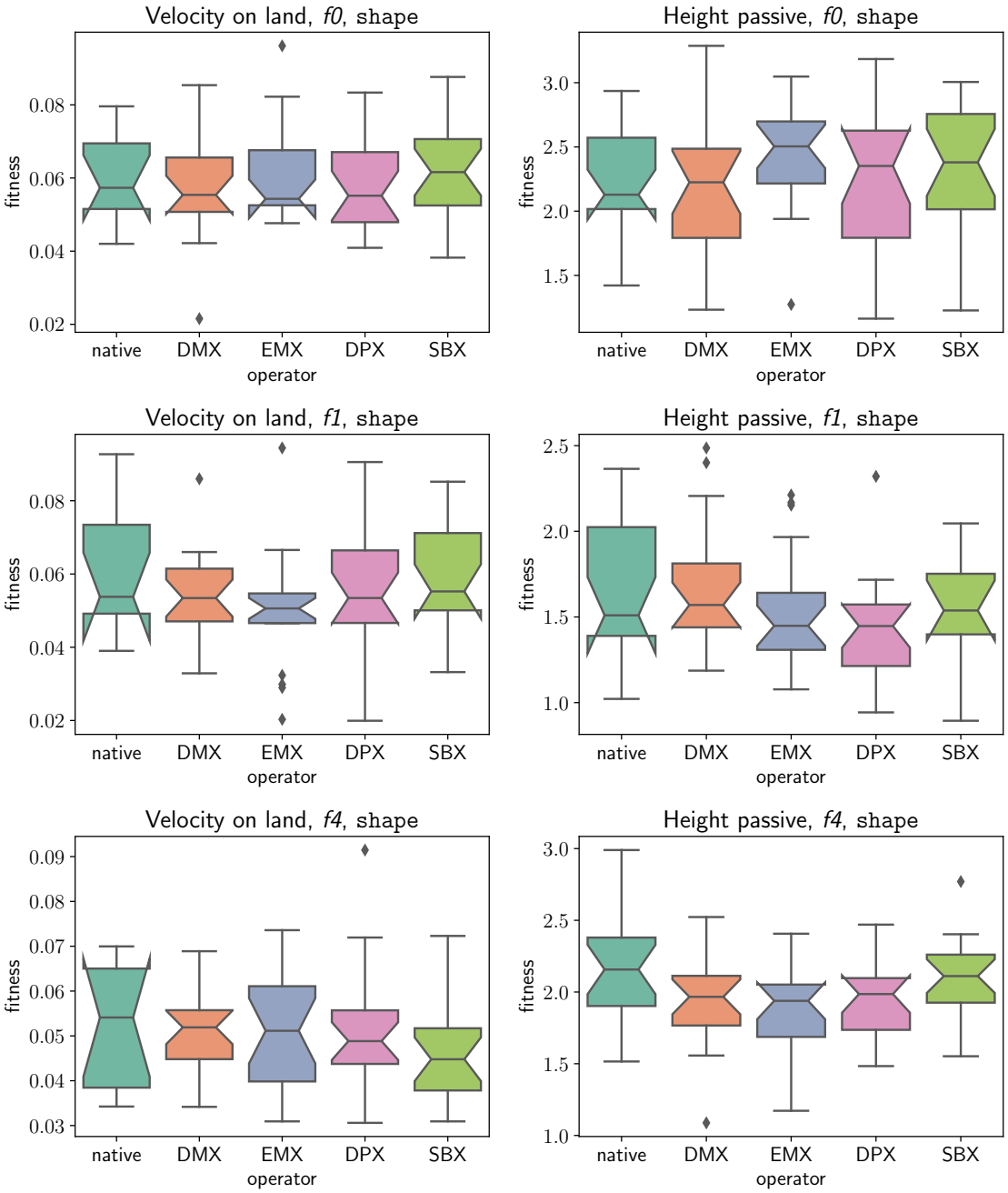


Figure G.1: Fitness of the best found solutions for the native crossover operator and DMX, EMX, DPX, and SBX operators using the opt measure for different optimization goals, and genetic representations. The boxplots show the distribution of the best fitness values from 20 runs for each combination of parameters. Rows: genetic encodings, columns: optimization objectives.

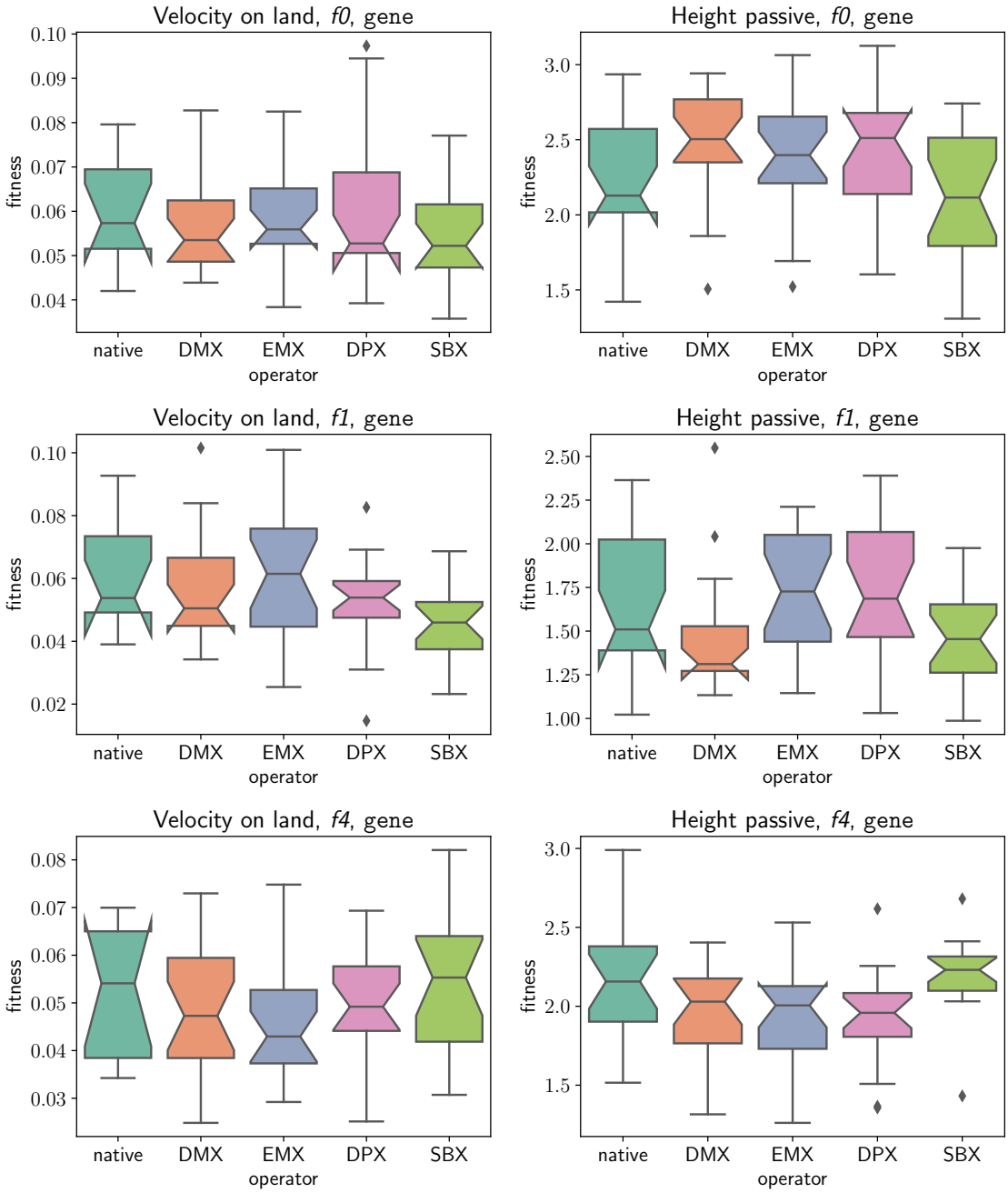


Figure G.2: Fitness of the best found solutions for the native crossover operator and DMX, EMX, DPX, and SBX operators using the opt measure for different optimization goals, and genetic representations. The boxplots show the distribution of the best fitness values from 20 runs for each combination of parameters. Rows: genetic encodings, columns: optimization objectives.

Bibliography

[1] A. L. Ames, D. R. Nadeau, and J. L. Moreland. *The VRML 2.0 sourcebook*. John Wiley & Sons, Inc., 1997.

[2] A. Arias-Montano, C. A. Coello Coello, and E. Mezura-Montes. Evolutionary algorithms applied to multi-objective aerodynamic shape optimization. *Computational optimization, methods and algorithms*, pages 211–240, 2011.

[3] F. G. Ashby. *Multidimensional models of perception and cognition*. Psychology Press, 2014.

[4] F. G. Ashby and N. A. Perrin. Toward a unified theory of similarity and recognition. *Psychological review*, 95(1):124, 1988.

[5] M. authors. Emscripten: a complete open source compiler toolchain to WebAssembly, 2020. URL: https://emscripten.org.

[6] G. Azamirad and B. Arezoo. Structural design of stamping die components using bi-directional evolutionary structural optimization method. *The International Journal of Advanced Manufacturing Technology*, 87:969–979, 2016.

[7] T. Back. Selective pressure in evolutionary algorithms: A characterization of selection mechanisms. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 57–62. IEEE, 1994. doi:10.1109/ICEC.1994.350042.

[8] T. Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[9] N. Bahramian and A. Khalkhali. Crashworthiness topology optimization of thin-walled square tubes, using modified bidirectional evolutionary structural optimization approach. *Thin-Walled Structures*, 147:106524, 2020.

[10] M. Balaji and V. Kamaraj. Evolutionary computation based multi-objective pole shape optimization of switched reluctance machine. *International Journal of Electrical Power & Energy Systems*, 43(1):63–69, 2012.

[11] A. Baldominos, Y. Saez, and P. Isasi. Evolutionary design of convolutional neural networks for human activity recognition in sensor-rich environments. *Sensors*, 18(4):1288, 2018.

[12] S. A. Barakat and S. Altoubat. Application of evolutionary global optimization techniques in the design of rc water tanks. *Engineering Structures*, 31(2):332–344, 2009.

[13] M. Barari, H. R. Karimi, and F. Razaghian. Analog circuit design optimization based on evolutionary algorithms. *Mathematical Problems in Engineering*, 2014, 2014.

[14] P. Baron, R. Fisher, A. Tuson, F. Mill, and A. Sherlock. A voxel-based representation for evolutionary shape optimization. *Ai Edam*, 13(3):145–156, 1999.

[15] A. Bataller, J. Cabrera, M. Clavijo, and J. Castillo. Evolutionary synthesis of mechanisms applied to the design of an exoskeleton for finger rehabilitation. *Mechanism and Machine Theory*, 105:31–43, 2016.

[16] P. J. Bentley. *Evolutionary Design by Computers*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1999.

[17] P. J. Bentley and J. P. Wakefield. The evolution of solid object designs using genetic algorithms. *Modern Heuristic Search Methods*, pages 199–215, 1996.

[18] P. J. Bentley and J. P. Wakefield. Generic evolutionary design. In *Soft computing in engineering design and manufacturing*, pages 289–298. Springer, 1998.

[19] M. Bilalić, P. McLeod, and F. Gobet. Why good thoughts block better ones: The mechanism of the pernicious einstellung (set) effect. *Cognition*, 108(3):652–661, 2008.

[20] K. D. Boese. *Cost versus distance in the traveling salesman problem*. UCLA Computer Science Department, 1995.

[21] N. Y. Bogula, S. Chermoshencev, and I. Suzdaltsev. Evolutionary algorithms for digital electronic printed circuit board design. In *2015 XVIII International Conference on Soft Computing and Measurements (SCM)*, pages 153–156. IEEE, 2015. doi:10.1109/SCM.2015.7190440.

[22] E. Bonabeau, S. Guérin, D. Snyers, P. Kuntz, and G. Theraulaz. Three-dimensional architectures grown by simple 'stigmergic' agents. *BioSystems*, 56(1):13–32, 2000.

[23] J. C. Bongard. The utility of evolving simulated robot morphology increases with task complexity for object manipulation. *Artificial life*, 16(3):201–223, 2010.

[24] J. C. Bongard, R. Pfeifer, et al. Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In *Proceedings of the genetic and evolutionary computation conference*, volume 829836, 2001.

[25] E. O. Brigham and R. E. Morrow. The fast fourier transform. *IEEE Spectrum*, 4(12):63–70, 1967. doi:10.1109/MSPEC.1967.5217220.

[26] L. Brodbeck, S. Hauser, and F. Iida. Morphological evolution of physical robots through model-free phenotype development. *PloS one*, 10(6):e0128444, 2015.

[27] T. Broughton, A. Tan, and P. Coates. The use of genetic programming in exploring 3D design worlds: A report of two projects by Msc students at CECA UEL. In *CAAD Futures 1997: Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures Held in Munich, Germany, 4–6 August 1997*, pages 885–915. Springer, 1997.

[28] J. Byrne. *Approaches to evolutionary architectural design exploration using grammatical evolution*. PhD thesis, University College Dublin, 2012.

[29] H. M. Cartwright and S. P. Harris. Analysis of the distribution of airborne pollution using genetic algorithms. *Atmospheric Environment. Part A. General Topics*, 27(12):1783–1791, 1993.

[30] O. Carugo. Statistical validation of the root-mean-square-distance, a measure of protein structural proximity. *Protein Engineering, Design & Selection*, 20(1):33–37, 2007. doi:10.1093/protein/gzl051.

[31] R. d. Carvalho, R. R. Saldanha, B. Gomes, A. C. Lisboa, and A. Martins. A multi-objective evolutionary algorithm based on decomposition for optimal design of Yagi-Uda antennas. *IEEE Transactions on Magnetics*, 48(2):803–806, 2012.

[32] P. Castillo, M. Arenas, J. Castillo-Valdivieso, J. Merelo, A. Prieto, and G. Romero. Artificial neural networks design using evolutionary algorithms. In *Advances in Soft Computing: Engineering Design and Manufacturing*, pages 43–52. Springer, 2003.

[33] T. Chabuk, J. Reggia, J. Lohn, and D. Linden. Causally-guided evolutionary optimization and its application to antenna array design. *Integrated Computer-Aided Engineering*, 19(2):111–124, 2012.

[34] K. Y. Chan, M. E. Aydin, and T. C. Fogarty. An epistasis measure based on the analysis of variance for the real-coded representation in genetic algorithms. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, volume 1, pages 297–304. IEEE, 2003.

[35] W. S. Chee and J. Teo. Simultaneous evolutionary-based optimization of controller and morphology of snake-like modular robots. In *2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology*, pages 37–42. IEEE, 2014.

[36] D. Chen, T. Aoki, N. Homma, T. Terasaki, and T. Higuchi. Graph-based evolutionary design of arithmetic circuits. *IEEE Transactions on Evolutionary Computation*, 6(1):86–100, 2002.

[37] O. Chocron. Evolutionary design of modular robotic arms. *Robotica*, 26(3):323–330, 2008.

[38] O. Chocron and P. Bidaud. Genetic design of 3D modular manipulators. In *Proceedings of International Conference on Robotics and Automation*, volume 1, pages 223–228. IEEE, 1997.

[39] C. A. C. Coello, A. D. Christiansen, and A. H. Aguirre. Towards automated evolutionary design of combinational circuits. *Computers & Electrical Engineering*, 27(1):1–28, 2000.

[40] C. C. Coello, F. S. Hernández, and F. A. Farrera. Optimal design of reinforced concrete beams using genetic algorithms. *Expert systems with Applications*, 12(1):101–108, 1997.

[41] J. Cohoon, J. Kairo, and J. Lienig. Evolutionary algorithms for the physical design of vlsi circuits. *Advances in evolutionary computing: theory and applications*, pages 683–711, 2003.

[42] M. Cook, S. Colton, and J. Gow. Initial results from co-operative co-evolution for automated platformer design. In *Applications of Evolutionary Computation: EvoApplications 2012: EvoCOMNET, EvoCOMPLEX, EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoNUM, EvoPAR, EvoRISK, EvoSTIM, and EvoSTOC, Málaga, Spain, April 11-13, 2012, Proceedings*, pages 194–203. Springer, 2012.

[43] F. Corucci, M. Calisti, H. Hauser, and C. Laschi. Novelty-based evolutionary design of morphing underwater robots. In *Proceedings of the 2015 annual conference on Genetic and Evolutionary Computation*, pages 145–152, 2015.

[44] T. F. Cox and M. A. A. Cox. *Multidimensional scaling*. Chapman and Hall/CRC, 2000.

[45] D. Dasgupta and Z. Michalewicz. *Evolutionary algorithms in engineering applications*. Springer Science & Business Media, 2013. doi:10.1007/978-3-662-03423-1.

[46] K. Deb and S. Gulati. Design of truss-structures for minimum weight using genetic algorithms. *Finite elements in analysis and design*, 37(5):447–465, 2001.

[47] K. Deb and S. Jain. Multi-speed gearbox design using multi-objective evolutionary algorithms. *J. Mech. Des.*, 125(3):609–619, 2003.

[48] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002. doi:10.1109/4235.996017.

[49] P. Della Vecchia, E. Daniele, and E. D'Amato. An airfoil shape optimization technique coupling parsec parameterization and evolutionary algorithm. *Aerospace Science and Technology*, 32(1):103–110, 2014.

[50] A. Dettmann, M. Roemmermann, and F. Cordes. Evolutionary development of an optimized manipulator arm morphology for manipulation and rover locomotion. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 2567–2573. IEEE, 2011.

[51] P. Di Barba. Evolutionary multiobjective optimization methods for the shape design of industrial electromagnetic devices. *IEEE Transactions on Magnetics*, 45(3):1436–1441, 2009. doi:10.1109/TMAG.2009.2012665.

[52] K. Diederichs. Structural superposition of proteins with unknown alignment and detection of topological similarity using a six-dimensional search algorithm. *Proteins: Structure, Function, and Bioinformatics*, 23(2):187–195, 1995.

[53] I. G. Dino. An evolutionary approach for 3D architectural space layout design exploration. *Automation in Construction*, 69:131–150, 2016.

[54] J. Doucette and M. I. Heywood. Novelty-based fitness: An evaluation under the Santa Fe Trail. In *European Conference on Genetic Programming*, pages 50–61. Springer, 2010.

[55] M. Ebner. Evolutionary design of objects using scene graphs. In *Genetic Programming: 6th European Conference, EuroGP 2003 Essex, UK, April 14–16, 2003 Proceedings 6*, pages 47–58. Springer, 2003.

[56] P. Eggenberger et al. Evolving morphologies of simulated 3D organisms based on differential gene expression. In *Proceedings of the fourth european conference on Artificial Life*, pages 205–213, 1997.

[57] S. C. Esquivel, H. A. Leiva, and R. H. Gallardt. Selection mechanisms in evolutionary algorithms. *Fundamenta Informaticae*, 35(1-4):17–33, 1998.

[58] A. Faíña, F. Bellas, F. López-Peña, and R. J. Duro. Edhmor: Evolutionary designer of heterogeneous modular robots. *Engineering Applications of Artificial Intelligence*, 26(10):2408–2423, 2013.

[59] D. B. Fogel. An overview of evolutionary programming. In *Evolutionary Algorithms*, pages 89–109. Springer, 1999.

[60] L. J. Fogel. *Biotechnology: Concepts and Applications*. Prentice-Hall, 1963.

[61] B. Freisleben and P. Merz. New genetic local search operators for the traveling salesman problem. *Parallel Problem Solving from Nature—PPSN IV*, pages 890–899, 1996. doi:10.1007/3-540-61723-X_1052.

[62] P. Funes and J. Pollack. Evolutionary body building: Adaptive physical designs for robots. *Artificial Life*, 4(4):337–357, 1998.

[63] V. J. Gan, H. Wong, K. T. Tse, J. C. Cheng, I. M. Lo, and C. M. Chan. Simulation-based evolutionary optimization for energy-efficient layout plan design of high-rise residential buildings. *Journal of cleaner production*, 231:1375–1388, 2019.

[64] A. Gandelli, F. Grimaccia, M. Mussetta, P. Pirinoli, and R. E. Zich. Genetical swarm optimization: an evolutionary algorithm for antenna design. *Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije*, 47(3-4):105–112, 2006.

[65] A. H. Gandomi, A. R. Kashani, D. A. Roke, and M. Mousavi. Optimization of retaining wall design using evolutionary algorithms. *Structural and multidisciplinary optimization*, 55:809–825, 2017.

[66] L. Giraud-Moreau and P. Lafon. A comparison of evolutionary algorithms for mechanical design components. *Engineering Optimization*, 34(3):307–322, 2002.

[67] M. N. Glibovets and N. M. Gulayeva. A review of niching genetic algorithms for multimodal function optimization. *Cybernetics and Sys. Anal.*, 49(6):815–820, November 2013. doi:10.1007/s10559-013-9570-8.

[68] A. Globus, J. Lawton, and T. Wipke. Automatic molecular design using evolutionary techniques. *Nanotechnology*, 10(3):290, 1999.

[69] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[70] D. E. Goldberg, J. Richardson, et al. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum, 1987.

[71] T. G. Gordon. Exploring models of development for evolutionary circuit design. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, volume 3, pages 2050–2057. IEEE, 2003.

[72] S. K. Goudos, A. Tsiflikiotis, D. Babas, K. Siakavara, C. Kalialakis, and G. K. Karagiannidis. Evolutionary design of a dual band e-shaped patch antenna for 5G mobile communications. In *2017 6th international conference on modern circuits and systems technologies (MOCAST)*, pages 1–4. IEEE, 2017.

[73] D. Greiner, J. J. Aznárez, O. Maeso, and G. Winter. Single-and multi-objective shape design of Y-noise barriers using evolutionary computation and boundary elements. *Advances in Engineering Software*, 41(2):368–378, 2010.

[74] Z. Guo and B. Li. Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system. *Frontiers of Architectural Research*, 6(1):53–62, 2017.

[75] R. W. Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

[76] F. Hamza, H. Abderazek, S. Lakhdar, D. Ferhat, and A. R. Yıldız. Optimum design of cam-roller follower mechanism using a new evolutionary algorithm. *The International Journal of Advanced Manufacturing Technology*, 99:1267–1282, 2018.

[77] K. Hamza and K. Saitou. Optimization of constructive solid geometry via a tree-based multi-objective genetic algorithm. In *Genetic and Evolutionary Computation–GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004. Proceedings, Part II*, pages 981–992. Springer, 2004.

[78] W. H. Hassan, Z. H. Attea, and S. S. Mohammed. Optimum layout design of sewer networks by hybrid genetic algorithm. *Journal of Applied Water Engineering and Research*, 8(2):108–124, 2020.

[79] R. L. Haupt. Antenna design with a mixed integer genetic algorithm. *IEEE Transactions on Antennas and Propagation*, 55(3):577–582, 2007.

[80] J. He, C. Reeves, C. Witt, and X. Yao. A note on problem difficulty measures in black-box optimization: Classification, realizations and predictability. *Evolutionary Computation*, 15(4):435–443, 2007.

[81] J. H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2):88–105, 1973. doi:10.1137/0202009.

[82] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[83] G. Hornby, A. Globus, D. Linden, and J. Lohn. Automated antenna design with evolutionary algorithms. In *Space 2006*, page 7242. 2006.

[84] G. S. Hornby. Creating complex building blocks through generative representations. In *Proceedings of the 2003 AAAI Spring Symposioum: Computational Synthesis: From Basic Building Blocks to High Level Functionality*, pages 98–105, 2003.

[85] G. S. Hornby. Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1729–1736. ACM, 2005.

[86] G. S. Hornby. Improving the scalability of generative representations for openended design. *Genetic Programming Theory and Practice V*, pages 125–142, 2008.

[87] G. S. Hornby, H. Lipson, and J. B. Pollack. Evolution of generative design systems for modular physical robots. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 4146–4151. IEEE, 2001.

[88] G. S. Hornby and J. B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 congress on evolutionary computation (ieee cat. no. 01th8546)*, volume 1, pages 600–607. IEEE, 2001.

[89] G. S. Hornby and J. B. Pollack. *Generative representations for evolutionary design automation*. Brandeis University, 2003.

[90] X. Huang and Y. Xie. Optimal design of periodic structures using evolutionary topology optimization. *Structural and Multidisciplinary Optimization*, 36:597–606, 2008.

[91] M. D. Huntington, L. J. Lauhon, and T. W. Odom. Subwavelength lattice optics by evolutionary design. *Nano letters*, 14(12):7195–7200, 2014.

[92] P. Husbands, G. Jermy, M. McIlhagga, and R. Ives. Two applications of genetic algorithms to component design. In *Evolutionary Computing: AISB Workshop Brighton, UK, April 1–2, 1996 Selected Papers*, pages 50–61. Springer, 1996. doi:doi.org/10.1007/BFb0032772.

[93] E. Iuliano and D. Quagliarella. Proper orthogonal decomposition, surrogate modelling and evolutionary optimization in aerodynamic design. *Computers & Fluids*, 84:327–350, 2013.

[94] N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani. Three-dimensional shape searching: state-of-the-art review and future trends. *Computer-Aided Design*, 37(5):509–530, 2005.

[95] P. Jaccard. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50, 1912.

[96] A. Jahangirian and A. Shahrokhi. Aerodynamic shape optimization using efficient evolutionary algorithms and unstructured CFD solver. *Computers & Fluids*, 46(1):270–276, 2011.

[97] A. Jain, R. K. Bhattacharjya, and S. Sanaga. Optimal design of composite channels using genetic algorithm. *Journal of Irrigation and Drainage Engineering*, 130(4):286–295, 2004.

[98] A. Jaszkiewicz and P. Kominek. Genetic local search with distance preserving recombination operator for a vehicle routing problem. *European Journal of Operational Research*, 151(2):352–364, 2003.

[99] A. Jaszkiewicz, P. Kominek, and M. Kubiak. Adaptation of the genetic local search algorithm to a car sequencing problem. In *7th National Conference on Evolutionary Algorithms and Global Optimization, Kazimierz Dolny, Poland*, pages 67–74, 2004.

[100] A. Javadian, N. Nariman-zadeh, and A. Jamali. Evolutionary design of marginally robust multivariable PID controller. *Engineering Applications of Artificial Intelligence*, 121:105938, 2023.

[101] M. M. Joly, T. Verstraete, and G. Paniagua. Integrated multifidelity, multidisciplinary evolutionary design optimization of counterrotating compressors. *Integrated Computer-Aided Engineering*, 21(3):249–261, 2014.

[102] E. A. Jones and W. T. Joines. Design of Yagi-Uda antennas using genetic algorithms. *IEEE Transactions on Antennas and Propagation*, 45(9):1386–1392, 1997.

[103] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[104] A. Joshi, S. Kale, S. Chandel, and D. K. Pal. Likert scale: Explored and explained. *Current Journal of Applied Science and Technology*, pages 396–403, 2015.

[105] M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. *Handbooks in operations research and management science*, 7:225–330, 1995.

[106] I. Karen, M. Yazici, and A. Shukla. Designing foam filled sandwich panels for blast mitigation using a hybrid evolutionary optimization algorithm. *Composite Structures*, 158:72–82, 2016.

[107] A. R. Kashani, M. Gandomi, C. V. Camp, and A. H. Gandomi. Optimum design of shallow foundation using evolutionary algorithms. *Soft Computing*, 24:6809–6833, 2020.

[108] L. Kaufman. Partitioning around medoids (program pam). *Finding groups in data*, 344:68–125, 1990.

[109] R. Kicinger, T. Arciszewski, and K. De Jong. Morphogenesis and structural design: cellular automata representations of steel structures in tall buildings. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, volume 1, pages 411–418. IEEE, 2004.

[110] R. Kicinger, T. Arciszewski, and K. DeJong. Evolutionary design of steel structures in tall buildings. *Journal of Computing in Civil Engineering*, 19(3):223–238, 2005.

[111] H.-S. Kim and S.-B. Cho. Application of interactive genetic algorithm to fashion design. *Engineering applications of artificial intelligence*, 13(6):635–644, 2000.

[112] M. Kiptiah binti Ariffin, S. Hadi, and S. Phon-Amnuaisuk. Evolving 3D models using interactive genetic algorithms and l-systems. In *Multi-disciplinary Trends in Artificial Intelligence: 11th International Workshop, MIWAI 2017, Gadong, Brunei, November 20-22, 2017, Proceedings 11*, pages 485–493. Springer, 2017.

[113] N. Kirchner-Bossi and F. Porté-Agel. Wind farm area shape optimization using newly developed multi-objective evolutionary algorithms. *Energies*, 14(14):4185, 2021.

[114] A. Klejda, M. Komosinski, and A. Mensfelt. Diversification techniques and distance measures in evolutionary design of 3D structures. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22)*, Boston, USA, 2022. ACM, ACM. doi:10.1145/3520304.3528948.

[115] M. Kociecki and H. Adeli. Shape optimization of free-form steel space-frame roof structures with complex geometries using evolutionary computing. *Engineering Applications of Artificial Intelligence*, 38:168–182, 2015.

[116] M. Komosinski. Applications of a similarity measure in the analysis of populations of 3D agents. *Journal of Computational Science*, 2017. doi:10.1016/j.jocs.2016.10.004.

[117] M. Komosinski. Artificial life and nature-inspired algorithms, 2023. Lecture script. URL: http://www.cs.put.poznan.pl/mkomosinski/lectures/MK_ArtLife.pdf.

[118] M. Komosinski and M. Kubiak. Quantitative measure of structural and geometric similarity of 3D morphologies. *Complexity*, 16(6):40–52, 2011. URL: http://www.framsticks.com/files/common/Komosinski_Kubiak_MeasureSimilarity3DMorphologies.pdf, doi:10.1002/cplx.20367.

[119] M. Komosinski and A. Mensfelt. A flexible dissimilarity measure for active and passive 3D structures and its application in the fitness–distance analysis. In P. Kaufmann and P. A. Castillo, editors, *Applications of Evolutionary Computation*. Springer, Springer, 2019. URL: http://www.framsticks.com/files/common/DissimilarityMeasure3DStructuresFitnessDistance.pdf, doi:10.1007/978-3-030-16692-2_8.

[120] M. Komosinski and A. Mensfelt. Human perception of similarity of 3D graph structures. Technical Report RA-07/2020, Poznan University of Technology, 2020.

[121] M. Komosinski and K. Miazga. Measuring properties of movement in populations of evolved 3D agents. In H. Fellermann, J. Bacardit, A. Goñi-Moreno, and R. M. Füchslin, editors, *Artificial Life Conference Proceedings*, pages 485–492. MIT Press, 2019. doi:10.1162/isal_a_00208.

[122] M. Komosinski and K. Rosinski. Estimating similarity of neural network dynamics. Technical Report RA-10/10, Poznan University of Technology, Institute of Computing Science, 2010. URL: http://www.framsticks.com/files/common/SimilarityNeuralNetworkDynamics.pdf.

[123] M. Komosiński and A. Rotaru-Varga. Comparison of different genotype encodings for simulated three-dimensional agents. *Artificial Life*, 7(4):395–418, 2001.

[124] M. Komosinski and S. Ulatowski. Framsticks: Creating and understanding complexity of life. In M. Komosinski and A. Adamatzky, editors, *Artificial Life Models in Software*, chapter 5, pages 107–148. Springer, London, 2nd edition, 2009.

[125] M. Komosinski and S. Ulatowski. Framsticks SDK (Software Development Kit), 2023. URL: http://www.framsticks.com/sdk.

[126] M. Komosinski and S. Ulatowski. Framsticks web site, 2023. URL: http://www.framsticks.com.

[127] M. Komosinski and S. Ulatowski. Python interface for Framsticks, 2023. URL: http://www.framsticks.com/trac/framsticks/browser/framspy.

[128] J. R. Koza et al. *Genetic programming II*, volume 17. MIT press Cambridge, 1994.

[129] K. Krawiec and P. Lichocki. Approximating geometric crossover in semantic space. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 987–994, 2009.

[130] P. Krčah. Towards efficient evolutionary design of autonomous robots. In *Evolvable Systems: From Biology to Hardware: 8th International Conference, ICES 2008, Prague, Czech Republic, September 21-24, 2008. Proceedings 8*, pages 153–164. Springer, 2008.

[131] P. Krčah. Solving deceptive tasks in robot body-brain co-evolution by searching for behavioral novelty. In *Advances in robotics and virtual reality*, pages 167–186. Springer, 2012.

[132] M. Kubiak. Systematic construction of recombination operators for the vehicle routing problem. *Foundations of Computing and Decision Sciences*, 29(3), 2004.

[133] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[134] Y. Kuwahara. Multiobjective optimization design of Yagi-Uda antenna. *Antennas and Propagation, IEEE Transactions on*, 53:1984 – 1992, 07 2005. doi:10.1109/TAP.2005.848501.

[135] E. M. W. Lameijer. *Interactive evolutionary algorithms and data mining for drug design*. PhD thesis, Leiden University, 2010.

[136] E.-W. Lameijer, J. N. Kok, T. Bäck, and A. P. IJzerman. The molecule evoluator. an interactive evolutionary algorithm for the design of drug-like molecules. *Journal of chemical information and modeling*, 46(2):545–552, 2006.

[137] R. Le Riche and R. Haftka. Improved genetic algorithm for minimum thickness composite laminate design. *Composites Engineering*, 5(2):143–161, 1995.

[138] D. Lee, C. Morillo, G. Bugeda, S. Oller, and E. Oñate. Multilayered composite structure design optimisation using distributed/parallel multi-objective evolutionary algorithms. *Composite structures*, 94(3):1087–1096, 2012.

[139] J. Lehman and K. O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI)*. MIT Press, 2008.

[140] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

[141] J. Lehman and K. O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, pages 211–218, 2011.

[142] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

[143] X. Li, M. G. Epitropakis, K. Deb, and A. Engelbrecht. Seeking multiple solutions: An updated survey on niching methods and their applications. *IEEE Transactions on Evolutionary Computation*, 21(4):518–538, 2016.

[144] Y. F. Li, X. Huang, F. Meng, and S. Zhou. Evolutionary topological design for phononic band gap crystals. *Structural and Multidisciplinary Optimization*, 54:595–617, 2016.

[145] Y. Lian, A. Oyama, and M.-S. Liou. Progress in design optimization using evolutionary algorithms for aerodynamic problems. *Progress in Aerospace Sciences*, 46(5-6):199–223, 2010.

[146] H. Lim and H. Kim. Multi-objective airfoil shape optimization using an adaptive hybrid evolutionary algorithm. *Aerospace Science and Technology*, 87:141–153, 2019.

[147] H. Lipson and J. B. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978, 2000.

[148] H. Lipson, V. Sunspiral, J. C. Bongard, and N. Cheney. On the difficulty of co-optimizing morphology and control in evolved virtual creatures. volume ALIFE 2016, the Fifteenth International Conference on the Synthesis and Simulation of Living Systems of *ALIFE 2023: Ghost in the Machine: Proceedings of the 2023 Artificial Life Conference*, pages 226–233, 07 2022. doi:10.1162/978-0-262-33936-0-ch042.

[149] B. Liu, H. Aliakbarian, Z. Ma, G. A. Vandenbosch, G. Gielen, and P. Excell. An efficient method for antenna design optimization based on evolutionary computation and machine learning techniques. *IEEE transactions on antennas and propagation*, 62(1):7–18, 2013.

[150] J. D. Lohn and S. P. Colombano. A circuit representation technique for automated circuit design. *IEEE Transactions on Evolutionary Computation*, 3(3):205–219, 1999.

[151] J. D. Lohn, W. F. Kraus, D. S. Linden, and A. Stoica. Evolutionary optimization of a quadrifilar helical antenna. 2:814–817, 2002.

[152] J. D. Lohn and J. A. Reggia. Discovery of self-replicating structures using a genetic algorithm. In *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, volume 2, pages 678–683. IEEE, 1995.

[153] Z. Lu, I. Whalen, Y. Dhebar, K. Deb, E. D. Goodman, W. Banzhaf, and V. N. Boddeti. Multiobjective evolutionary design of deep convolutional neural networks for image classification. *IEEE Transactions on Evolutionary Computation*, 25(2):277–291, 2020.

[154] A. G. Maldonado, J. Doucet, M. Petitjean, and B.-T. Fan. Molecular similarity and diversity in chemoinformatics: from theory to applications. *Molecular diversity*, 10:39–79, 2006.

[155] P. E. McKight and J. Najab. Kruskal-wallis test. *The corsini encyclopedia of psychology*, pages 1–1, 2010.

[156] J. F. Miller, P. Thomson, and T. Fogarty. Designing electronic circuits using evolutionary algorithms. Arithmetic circuits: A case study. *Genetic algorithms and evolution strategies in engineering and computer science*, pages 105–131, 1997.

[157] H. Monsef, M. Naghashzadegan, A. Jamali, and R. Farmani. Comparison of evolutionary multi objective optimization algorithms in optimum design of water distribution network. *Ain Shams Engineering Journal*, 10(1):103–111, 2019.

[158] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[159] S. Nagendra, D. Jestin, Z. Gürdal, R. T. Haftka, and L. T. Watson. Improved genetic algorithm for the design of stiffened composite panels. *Computers & Structures*, 58(3):543–555, 1996.

[160] G. Nicosia, S. Rinaudo, and E. Sciacca. An evolutionary algorithm-based approach to robust analog circuit design using constrained multi-objective optimization. In *Research and Development in Intelligent Systems XXIV: Proceedings of AI-2007, the Twenty-seventh SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 7–20. Springer, 2008.

[161] S. Obayashi. Multidisciplinary design optimization of aircraft wing planform based on evolutionary algorithms. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, volume 4, pages 3148–3153. IEEE, 1998.

[162] S. Obayashi, D. Sasaki, Y. Takeguchi, and N. Hirose. Multiobjective evolutionary computation for supersonic wing-shape optimization. *IEEE transactions on evolutionary computation*, 4(2):182–187, 2000.

[163] H. Ogawa and R. R. Boyce. Nozzle design optimization for axisymmetric scramjets by using surrogate-assisted evolutionary algorithms. *Journal of Propulsion and Power*, 28(6):1324–1338, 2012.

[164] M. O'Neill, J. McDermott, J. M. Swafford, J. Byrne, E. Hemberg, A. Brabazon, E. Shotton, C. McNally, and M. Hemberg. Evolutionary design using grammatical evolution and shape grammars: Designing a shelter. *International Journal of Design Engineering*, 3(1):4–24, 2010.

[165] M. O'Neill, J. M. Swafford, J. McDermott, J. Byrne, A. Brabazon, E. Shotton, C. McNally, and M. Hemberg. Shape grammars and grammatical evolution for evolutionary design. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1035–1042, 2009.

[166] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Matching 3D models with shape distributions. In *Proceedings of the International Conference on Shape Modeling and Applications*, SMI '01, page 154, USA, 2001. IEEE Computer Society. doi:10.1109/SMA.2001.923386.

[167] U.-M. O'Reilly and G. Ramachandran. A preliminary investigation of evolution as a form design strategy. In *Artificial Life VI*, pages 443–447. Los Angeles, CA, UAS, 1998.

[168] C. Paul and J. C. Bongard. The road less travelled: Morphology in the optimization of biped robot locomotion. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 1, pages 226–232. IEEE, 2001.

[169] M. Peysakhov and W. C. Regli. Using assembly representations to enable evolutionary design of Lego structures. *Ai Edam*, 17(2):155–168, 2003.

[170] N. Pholdee and S. Bureerat. Performance enhancement of multiobjective evolutionary optimisers for truss design using an approximate gradient. *Computers & structures*, 106:115–124, 2012.

[171] D. Pisinger and P. Toth. Knapsack problems. *Handbook of Combinatorial Optimization: Volume1–3*, pages 299–428, 1998. doi:10.1007/978-1-4613-0303-9_5.

[172] E. Pitzer and M. Affenzeller. A comprehensive survey on fitness landscape analysis. In *Recent Advances in Intelligent Engineering Systems*, pages 161–191. Springer, 2012.

[173] J. M. Ponce-Ortega, M. Serna-González, and A. Jiménez-Gutiérrez. Use of genetic algorithms for the optimal design of shell-and-tube heat exchangers. *Applied Thermal Engineering*, 29(2-3):203–209, 2009.

[174] P. Prusinkiewicz, M. Hammel, J. Hanan, and R. Mech. L-systems: from the theory to visual models of plants. In *Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences*, volume 3, pages 1–32, 1996.

[175] J. K. Pugh, L. B. Soros, and K. O. Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

[176] S. N. Qasem, S. M. Shamsuddin, and A. M. Zain. Multi-objective hybrid evolutionary algorithms for radial basis function neural network design. *Knowledge-Based Systems*, 27:475–497, 2012.

[177] S. Rajan. Sizing, shape, and topology design optimization of trusses using genetic algorithm. *Journal of structural engineering*, 121(10):1480–1487, 1995.

[178] S. Rajeev and C. Krishnamoorthy. Genetic algorithms-based methodologies for design optimization of trusses. *Journal of structural engineering*, 123(3):350–358, 1997.

[179] S. Rajeev and C. Krishnamoorthy. Genetic algorithm–based methodology for design optimization of reinforced concrete frames. *Computer-Aided Civil and Infrastructure Engineering*, 13(1):63–74, 1998.

[180] B. R. Rao and R. Tiwari. Optimum design of rolling element bearings using genetic algorithms. *Mechanism and machine theory*, 42(2):233–250, 2007.

[181] I. Rechenberg. Evolutionstrategie (evolution strategy). *Frommann-Holzboog, Stuttgart*, 1973.

[182] J. Reisinger, K. O. Stanley, and R. Miikkulainen. Towards an empirical measure of evolvability. In *Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation*, GECCO '05, page 257–264, New York, NY, USA, 2005. Association for Computing Machinery. doi:10.1145/1102256.1102315.

[183] G. Ren, J. Smith, J. Tang, and Y.-M. Xie. Underground excavation shape optimization using an evolutionary procedure. *Computers and Geotechnics*, 32(2):122–132, 2005.

[184] S. Risi, C. E. Hughes, and K. O. Stanley. Evolving plastic neural networks with novelty search. *Adapt Behav*, 18(6):470–491, December 2010. doi:10.1177/1059712310379923.

[185] M. Rommerman, D. Kuhn, and F. Kirchner. Robot design for space missions using evolutionary computation. In *2009 IEEE Congress on Evolutionary Computation*, pages 2098–2105. IEEE, 2009.

[186] M. A. Rosenman. *The Generation of Form Using an Evolutionary Approach*, pages 69–85. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. doi:10.1007/978-3-662-03423-1_4.

[187] F. Rothlauf. *Representations for genetic and evolutionary algorithms*. Springer, 2006. doi:10.1007/3-540-32444-5.

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

[189] R. Saravanan, S. Ramabalan, N. G. R. Ebenezer, and C. Dharmaraja. Evolutionary multi criteria design optimization of robot grippers. *Applied Soft Computing*, 9(1):159–172, 2009.

[190] B. Sareni and L. Krahenbuhl. Fitness sharing and niching methods revisited. *IEEE transactions on Evolutionary Computation*, 2(3):97–106, 1998.

[191] K. Sastry, D. E. Goldberg, and G. Kendall. Genetic algorithms. In *Search methodologies*, pages 93–117. Springer, 2014.

[192] M. Schoenauer. Shape representations and evolution schemes. In *Fifth Annual Conference on Evolutionary Programming*, 1996.

[193] M. S. Selig and V. L. Coverstone-Carroll. Application of a genetic algorithm to wind turbine design. *Journal of Energy Resources Technology*, 118(1):22–28, 1996.

[194] Y. Shao, B. Lu, H. Ou, and J. Chen. A new approach of preform design for forging of 3D blade based on evolutionary structural optimization. *Structural and Multidisciplinary Optimization*, 51:199–211, 2015.

[195] P. Shiakolas, D. Koladiya, and J. Kebrle. Optimum robot design based on task specifications using evolutionary techniques and kinematic, dynamic, and structural constraints. *Inverse Problems in Engineering*, 10(4):359–375, 2002.

[196] K. Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22, 1994.

[197] R. Smith. Open Dynamics Engine (ODE), 2017. URL: http://www.ode.org/.

[198] T. Smith, P. Husbands, and M. O'Shea. Fitness landscapes and evolvability. *Evolutionary computation*, 10(1):1–34, 2002.

[199] G. Soremekun, Z. Gürdal, R. Haftka, and L. Watson. Composite laminate design optimization by genetic algorithm with generalized elitist selection. *Computers & structures*, 79(2):131–143, 2001.

[200] W. M. Spears et al. Crossover or mutation. *Foundations of genetic algorithms*, 2:221–237, 1993.

[201] P. F. Stadler. Landscapes and their correlation functions. *Journal of Mathematical chemistry*, 20(1):1–45, 1996.

[202] S.-D. Stan, M. Manic, V. Maties, and R. Balan. Evolutionary approach to optimal design of 3 dof translation exoskeleton and medical parallel robots. In *2008 Conference on Human System Interactions*, pages 720–725. IEEE, 2008.

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

[204] K. O. Stanley and R. Miikkulainen. Efficient reinforcement learning through evolving neural network topologies. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 569–577. Morgan Kaufmann Publishers Inc., 2002.

[205] A. Stoica, G. Klimeck, C. Salazar-Lazaro, D. Keymeulen, and A. Thakoor. Evolutionary design of electronic devices and circuits. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1271–1278. IEEE, 1999. doi:10.1109/CEC.1999.782588.

[206] X. Sun, J. Yang, Y. Xie, X. Huang, and Z. Zuo. Topology optimization of composite structure using bi-directional evolutionary structural optimization method. *Procedia Engineering*, 14:2980–2985, 2011.

[207] J. Talamini, E. Medvet, and S. Nichele. Criticality-driven evolution of adaptable morphologies of voxel-based soft-robots. *Frontiers in Robotics and AI*, 8:673156, 2021.

[208] I. Tanev, T. Ray, and A. Buller. Automated evolutionary design, robustness, and adaptation of sidewinding locomotion of a simulated snake-like robot. *IEEE Transactions on Robotics*, 21(4):632–645, 2005.

[209] K.-S. Tang, K.-F. Man, S. Kwong, and Z.-F. Liu. Design and optimization of IIR filter structure using hierarchical genetic algorithms. *IEEE Transactions on Industrial Electronics*, 45(3):481–487, 1998.

[210] Y. Tang, A. Kurtz, and Y. F. Zhao. Bidirectional evolutionary structural optimization (BESO) based design method for lattice structure to be fabricated by additive manufacturing. *Computer-Aided Design*, 69:91–101, 2015.

[211] A. Thompson. Evolving fault tolerant systems. In *1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, pages 524–529. IET, 1995.

[212] V. Toğan and A. T. Daloğlu. Optimization of 3D trusses with adaptive approach in genetic algorithms. *Engineering Structures*, 28(7):1019–1027, 2006.

[213] R. Toledo, J. Aznárez, O. Maeso, and D. Greiner. Optimization of thin noise barrier designs using evolutionary algorithms and a dual BEM formulation. *Journal of Sound and Vibration*, 334:219–238, 2015.

[214] D. Tuhus-Dubrow and M. Krarti. Genetic-algorithm based approach to optimize building envelope design for residential buildings. *Building and environment*, 45(7):1574–1581, 2010.

[215] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977. doi:10.1037/0033-295X.84.4.327.

[216] S. Ulatowski. Framsticks simulation, 2023. URL: https://www.framsticks.com/files/presentations/sz_simulation.pdf.

[217] M. Vasundara and K. Padmanaban. Recent developments on machining fixture layout design, analysis, and optimization using finite element method and evolutionary techniques. *The International Journal of Advanced Manufacturing Technology*, 70:79–96, 2014.

[218] V. Venkatasubramanian, K. Chan, and J. M. Caruthers. Evolutionary design of molecules with desired properties using the genetic algorithm. *Journal of Chemical Information and Computer Sciences*, 35(2):188–195, 1995.

[219] H. Wang, J. Doherty, and Y. Jin. Hierarchical surrogate-assisted evolutionary multi-scenario airfoil shape optimization. In *2018 IEEE congress on evolutionary computation (CEC)*, pages 1–8. IEEE, 2018.

[220] H. Wang, J. Liu, and G. Wen. An efficient evolutionary structural optimization method for multi-resolution designs. *Structural and Multidisciplinary Optimization*, 62:787–803, 2020.

[221] J. Wang, M. Wang, M. Li, J. Xia, and Y. Dai. Multi-objective optimization design of condenser in an organic rankine cycle for low grade waste heat recovery using evolutionary algorithm. *International communications in heat and mass transfer*, 45:47–54, 2013.

[222] Y. Wang, B. Li, and Y. Chen. Digital IIR filter design using multi-objective optimization evolutionary algorithm. *Applied Soft Computing*, 11(2):1851–1857, 2011.

[223] S. Wannarumon and E. L. Bohez. A new aesthetic evolutionary approach for jewelry design. *Computer-Aided Design and Applications*, 3(1-4):385–394, 2006.

[224] J. Wernecke. *The Inventor Mentor: Programming Object-Oriented 3d Graphics with Open Inventor, Release 2*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 1993.

[225] J. P. Wieczorek, O. Gol, and Z. Michalewicz. An evolutionary algorithm for the optimal design of induction motors. *IEEE Transactions on Magnetics*, 34(6):3882–3887, 1998.

[226] S. S. Wong and K. C. Chan. Evoarch: An evolutionary algorithm for architectural layout design. *Computer-Aided Design*, 41(9):649–667, 2009.

[227] Y. Yu and Y. Xinjie. Cooperative coevolutionary genetic algorithm for digital IIR filter design. *IEEE Transactions on Industrial Electronics*, 54(3):1311–1318, 2007.

[228] X. Zhang, Y. M. Xie, and S. Zhou. A nodal-based evolutionary optimization algorithm for frame structures. *Computer-Aided Civil and Infrastructure Engineering*, 38(3):288–306, 2023.

[229] S. Y. Zheng, S. H. Yeung, W. S. Chan, K. F. Man, and K. S. Tang. Design of broadband hybrid coupler with tight coupling using jumping gene evolutionary algorithm. *IEEE Transactions on Industrial Electronics*, 56(8):2987–2991, 2009.

[230] Y. Zheng, Z. Y. Dong, Y. Xu, K. Meng, J. H. Zhao, and J. Qiu. Electric vehicle battery charging/swap stations in distribution systems: Comparison study and optimal planning. *IEEE Transactions on Power Systems*, 29(1):221–229, 2014. doi:10.1109/TPWRS.2013.2278852.

[231] Z. Zhuang, Y. M. Xie, Q. Li, and S. Zhou. Body-fitted bi-directional evolutionary structural optimization using nonlinear diffusion regularization. *Computer Methods in Applied Mechanics and Engineering*, 396:115114, 2022.

[232] A. Zolfaghari, M. Goharimanesh, and A. A. Akbari. Optimum design of straight bevel gears pair using evolutionary algorithms. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 39:2121–2129, 2017.

[233] V. Zykov, J. C. Bongard, and H. Lipson. Evolving dynamic gaits on a physical robot. In *Proceedings of Genetic and Evolutionary Computation Conference, Late Breaking Paper, GECCO*, volume 4, page 2004, 2004.