University Of Amsterdam

Master Thesis

Evolution of Soft Robots by Novelty Search

By:
Georgios Methenitis

Supervisors:
Arnoud Visser (UvA)
Dario Izzo (ESA)
Daniel Hennes (ESA)

October 2014

UNIVERSITY OF AMSTERDAM

Abstract

Faculty of Science
Artificial Intelligence

Master of Science

Evolution of Soft Robots by Novelty Search

by Georgios Methenitis

Soft robotics is a vivid research field on the science and engineering aspects of soft materials in mobile machines. Recent development in soft robotics and evolutionary optimization have shown the possibility to simultaneously evolve the morphology and locomotion of soft robots. Generative encoding coupled with neural evolution of augmented topologies shows promising results. Novelty search, unlike traditional optimization methods does not aim to optimize the objective but instead looks for novelty. Novelty search rewards diversity and leads to a boundless variety of solutions, mimicking natural evolution. Apart from the performance comparison between novelty and fitness based search, this thesis shows that new locomotion patterns can be produced by the former. Different types of selection algorithms for fitness and novelty based evolution are studied, as well as, a method to combine both is proposed. Finally, the performance objective-wise is tested under variant gravity conditions leading into a taxonomy of possible locomotion strategies given different gravity levels.

Acknowledgements

...

Contents

List of Figures

List of Tables

List of Algorithms

Dedicated to my...

Chapter 1

Introduction

Soft robotics is a field of research inspired by soft-bodied organisms, whereas the engineering and designing aspects of soft-structures are in the center of interest. Soft-robotics can make the interaction between robots and living organisms safe. In addition, it allows soft robots to function in more natural and complex environments, where rigid robots have disadvantages. Actuated soft materials that react to environmental changes, add complexity to the designing phase of soft-robot engineering, since the infinite degrees of freedom of soft structures and the possible distributions of materials, make the number of possibilities vast.

Approaching such a deep search space, is a heavy task. Recent developments in evolutionary optimization though, have shown the possibility of successful evolution of both the morphology and the locomotion strategy for soft robotic structures, where the genotype representation is of vital importance to the evolution. Generative encoding has shown promising results especially in specific problem domains, such as evolving controllers for robot gait and morphology evolution. As direct encoding provides a direct mapping from genotype to phenotype level, indirect determines a set of rules, functions that can be queried and generate each individual in the space of the phenotype. Recent work has proved that evolutionary methods coupled with a generative encoding genotype representation can actually evolve both the morphology and the locomotion behavior of soft-robotics in a virtual simulation.

Traditional evolutionary methods in pursuance of the objective function, defined by the user, are blind to generate enough diversity within the population, often driving the evolution towards local optima. Novelty search, unlike traditional optimization methods does not aim to optimize individuals towards an objective, but instead, looks for novelty. Novelty search rewards diversity and leads to a boundless variety of solutions, mimicking

natural evolution in such a way. Doing so, it has proven to be a successful method for searching vast spaces where the objective function is deceptive.

Soft-robotic structures have no limitations to the extent of possible morphologies that can be discovered. Within the same context, gravity conditions when robot locomotion strategies are investigated might be more decisive when it comes to the morphology of the robotic explorers. With the freedom soft-structures give to evolutionary techniques in respect to the designing part of the evolution (morphology), it is of interest to validate that a taxonomy of different locomotion strategies can be applied when the gravity acceleration varies. As these structures can be made completely out of soft materials, all constraints about the applicable shapes seize to exist, giving the opportunity to human or algorithms to design unconventional shapes.

1.1 Thesis Contribution

This thesis explores possibles ways of evolving the morphology and the locomotion strategy of soft structures under a virtual simulation environment. As baseline, an initial experiment is performed to confirm that these problems cannot be captured by a simple genetic method with direct encoding representation of genotype. Both direct and indirect encoding are also used under a random robot generator which shows the advantages of the latter in the produced structures, as well as, points out the need of a generative way to explore and exploit the geometrical properties of the problem. This encoding scheme is used paired to an evolutionary algorithm to verify results of previous work on the same domain, showing that generative representations for the genotype can indeed benefit these kind of evolutionary optimization methods. In addition, this thesis is exploring the effect of diversity based evolution can have in the performance of the evolved morphologies. Novelty search, a method rewarding the "new" in the behavior level is used for this purpose showing that not only same or better performance can be achieved through this method but also the diversity of behaviors is remarkably increased. Last, both search methods are used to evolve structures for a variety of gravity levels, expecting to show a different taxonomy of locomotion patterns under different conditions apart from general implications regarding the effect of gravity in the locomotion success of mobile machines.

1.2 Thesis Outline

Chapter 2, provides some background information on the field of soft robotics, an introduction to genetic algorithms, different encoding techniques for the genotype representation, neuroevolution algorithms, and finally, objective driven search is presented and compared to novelty search. In Chapter 3, related material about evolutionary techniques used to evolve artificial life, as well as the evolution of soft-robots morphology and locomotion are presented. Chapter 4, is a comprehensive documentation presenting details of the implementation of different evolutionary techniques. Chapter 5, gives a detailed presentation of the results achieved under different experimental setups. Next, in chapter 6, future applications and extensions of this work are provided. Chapter 7, serves as an epilogue to this thesis, where the impact of the contributions are discussed.

Chapter 2

Background

This chapter gives an short overview of the state-of-the-art in evolutionary algorithms, to introduce the concepts needed for the approach of this thesis. First, evolutionary algorithms and robotics are discussed in depth. More specifically, genetic algorithms are presented, the role of the encoding in the representation within an evolutionary setting, how artificial neural networks (ANNs) can represent an organism in evolutionary algorithms (EAs), and how these ANNs can be evolved when coupled with an EA. As part of the different encoding schemes, an indirect coding called compositional pattern producing networks is also discussed in detail. The aspect of the objective function in such evolutionary problems and the affect that has in the performance of the methods. Additionally, a search that uses an objective function that rewards diversity in the evolution is presented. Last, a field of robotics and material sciences, soft robotics is investigated in conjunction with ways that these soft material structures can be evolved and simulated in virtual environments, as well as already designed soft-robots in real life applications.

2.1 Evolutionary Algorithms

Evolutionary algorithms (EA) is a part of the evolutionary computation field, where generic population-based optimization algorithms are studied. The main idea is that an evolution is held and propagate from generation to generation holding a number of candidate solutions. These candidates are propagated within generations until a good solution is found or a maximum number of iterations has passed. One of the most important advantages of EAs is that they can approximate good solutions in very difficult optimization problems, where analytic methods cannot be applied. This non-deterministic way of evolution starting from random sampled candidates from the solution space, results that

the algorithm will come up with different results on separate runs. Another important fact of EAs is that holding a population of solutions can help avoiding being "trapped" in local optima of a specific function. The way EAs propagate from one generation to the other is simply by using all/part of the current candidate solutions to produce the next generation of them. Nevertheless, in evolutionary algorithms, the objective function trying to optimize has always meaning, only in respect to other individuals within the population, and not to the ultimate objective of each optimization problem.

2.1.1 Genetic Algorithms

Genetic are in principle part of the evolutionary algorithms and follow the same principles.

> Genetic algorithms are probabilistic search procedures designed to work on large problem spaces involving states that can be represented by strings.

Considering the above quote by [1], a genetic algorithm is a process of evolving a stream of values, which is a single solution in a high dimensional problem space. These values can be at their simplest form, bit $(0, 1)$, integer, or float values.

Each of these candidate solutions, is called a *phenotype*, and the stream the solution is derived from, *genotype* or *chromosome*. Genetic algorithms are also a type of evolution based on generations. Each generation holds a population of a fixed number of individuals which are initially randomly selected out of a distribution in the solution space. The iterative process that follows and creates, given the current population of individual genotypes, the new population, is called *generation*. Usually the algorithm terminates after a fixed step of generations or when the goal has been reached.

The way the next generation's population is produced completely depends on the current population, genotypes are selected to breed new individuals. There two basic ways for a new genotype to be produced. The first way is called *mutation* and requires only one individual from the current population. Mutation will change one or more values in the *chromosome* of the selected individual to create a new one and maintain the genetic diversity from one generation to the other. *Crossover* is the second basic genetic operator and requires two or more parents for each new individual. This operator is similar to biological crossover and it uses parts from all parents to create a new chromosome.

The way individuals are selected after each successful generation in order to produce new individuals, belongs to the process of *selection*. Selection, as the name reveals itself, selects which individuals will become parents and which individuals will not. The

selection criteria, as it also happen in some natural environments where the most fit organisms survive, is a function that can approximate how good is an individual. This *objective* function, also called *fitness*, is a measure of how good an individual is (i.e. total displacement of a robot's body while trying to evolve walking). With the knowledge of the fitness function added to the evolution, weak individuals are most of the times discarded from the breeding process. Selecting parents randomly from the top part of the population or selecting parent via *tournament*, known also as *competition* (i.e., multiple random picks of individuals from the whole generation's population keeping the best pick) are two of the basic selection methods in evolutionary algorithms.

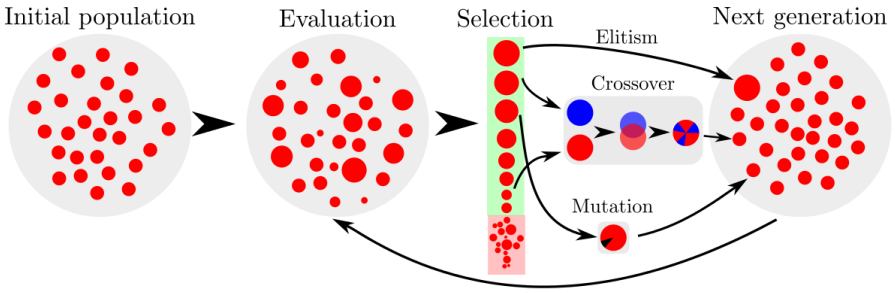


FIGURE 2.1: Basic pipeline of an evolutionary method.

Figure 2.1, illustrates the algorithmic pipeline of an evolutionary method, as described above. The pipeline starts with an random initialized population which is then evaluated (size refers to how "good" is each individual). The selection process is then following, whereas individuals are sorted based on their goodness in respect to the target of the evolution, and a set of the best is selected to produce the next generation where with the help of the three main genetic operators, elitism, crossover and mutation. The next generation will also be evaluated in respect to the same objective function and the iterative process will continue.

2.2 Evolutionary Robotics

Evolutionary robotics [2] (ER) is a method that makes use of evolutionary computation algorithms to develop (evolve) robot morphology or robot controllers, without the direct design-programming by human engineers. Most research concentrate on developing robot controllers. One big advantage of this method is that it can evolve controllers for robots for environments that human designers and engineers do not have enough knowledge about (i.e., designing a robot controller for another planet, where surface type and gravity level might be crucial variables for the design of an exploring robot). In the same fashion as natural evolution, evolutionary techniques work with a population of random initialized controllers. The candidate population individuals (robot controllers)

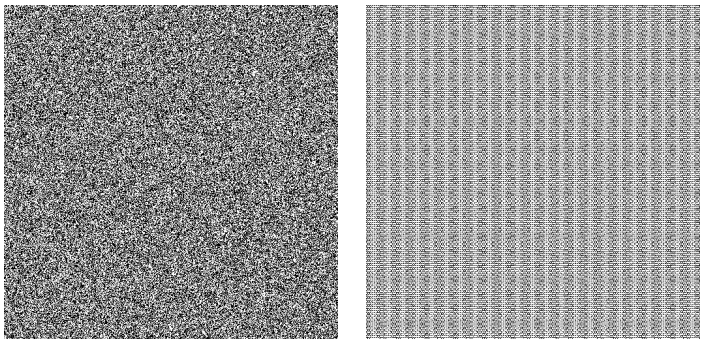


FIGURE 2.2: Comparison *direct* encoding versus *generative* for the binary image example.

used in ER applications may be drawn from some subset of the set of artificial neural networks (ANNs), whereas simpler versions of genetic algorithm applications use `bit`-streams that directly map the controller. The controllers in the better performing robots are selected then, altered and propagated through mutation, crossover, and other genetic operations, in a repeating process that mimics natural evolution. Evolutionary robotics is done with many different objectives, often at the same time. These include creating useful controllers for real-world robot tasks, reproducing biological phenomena, etc.. Creating controllers via artificial evolution requires a large number of evaluations of a large population. This is usually takes a lot of computational time, which is one of the reasons why evolution of such controllers is usually evaluated within a simulation software. Also, initial random controllers may exhibit potentially harmful behavior, such as repeatedly crashing the robot into a wall, which may damage a physical robot.

Apart from evolutionary methods to develop robot controllers reinforcement learning is used, rewarding actions, which translates to state-action pairs that lead to high rewarding behaviors, as a result, a robot controller can be indirectly built. Applying evolved robot controllers to real robots in a physical environment is an extremely difficult task, since simulators in front of the limitations of computing efficiency sacrifice the accuracy. In some cases, evolutionary methods can be used to design the physical structure (morphology) of the robot [3], in addition or in place of the controller. This thesis is exploring that aspect of evolution. Developmental robotics is a field related to evolutionary robotics, while instead of evolving through generations towards more fit controllers, it is trying to mimic life-like learning starting from a "blank" state in which the robot's "brain" is initialized and everything is unknown.

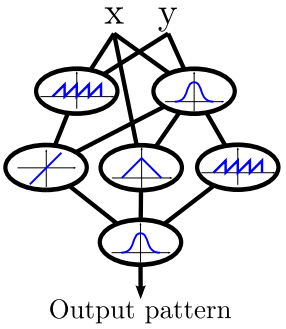


FIGURE 2.3: Compositional pattern producing networks have identical network structure with artificial neural networks.

2.3 Direct-Indirect Encoding of the Genotype

A simple direct encoding was described in the previous section, when a single dimension stream of bits or numbers describe the chromosome. When the dimensions of the task define the length of this genome, we have a *direct* encoding, which means that the genotype-phenotype mapping is a straightforward function. An example of this encoding could be the design of a two dimensional binary image. In direct encoding the genotype of this picture can be represented by a stream of bits which has the same length as the image's pixels. In other cases, when there is no direct mapping between the genotype and the phenotype, indirect encoding can be presented, whereas a set of rules or a function can map the phenotype from the genotype. In cases that the phenotype space can be represented by a Cartesian n-dimensional space, an indirect encoded chromosome can be a function that is queried for each coordinate in a specific resolution and represents the phenotype. For the same binary image example, indirect encoding the genotype would be represented by a function that gives pixel values 0 or 1 for every pixel's coordinate.

Figure 2.2, illustrates the difference between direct and indirect encoding, an example binary image is shown for both encoding schemes, in the first case (direct) the genotype is a binary stream which length is the equal with the number of pixels producing the value of each pixel directly. The latter encoding, uses a genome of length 3, as many as the coefficients of a linear combination of the function that involves sin, cos, tan-functions, the result is taken after applying the same function for each pixel coordinate. Even when a simple function is used the phenotype holds some of its functions properties such as symmetry and repetition, resulting in a pattern that direct encoding cannot produce.

2.3.1 Compositional Pattern Producing Networks

Encoding plays an important role and it is critical about the performance of evolutionary algorithms ecpecially when huge problem spaces are present. Research have shown

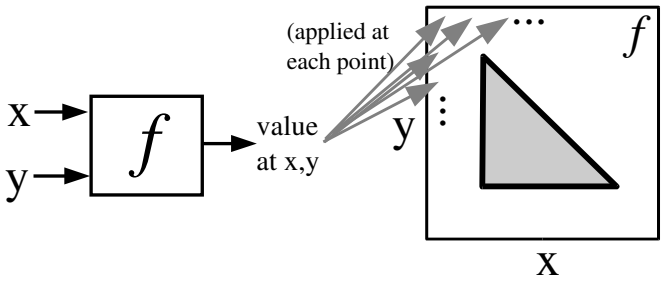


FIGURE 2.4: CPPNs work as a function f that is being queried for the whole n-dimensional Cartesian space in which space maps the phenotype, in this case the phenotype is the triangle, figure taken by [6].

that the genotype-phenotype mapping can affect performance [4]. More than that, the geometrical implications of the problem also have some potentially important role in the encoding problem. Thus, the role of symmetry to machine learning technique is important [5], especially in applications like board games, robot controllers, biped walking, etc., where geometric regularities can be descriptive about the nature of the problem.

Compositional pattern producing networks [6] or CPPNs are artificial neural networks in their base with an extended set of activation functions (fig. 2.3). Results by this encoding show that repetitive and structural patterns can be produced in this generative mapping from the genotype to the phenotype space. Just like in the previous two dimensional image representation of a phenotype, CPPNs generate phenotypes that can be interpreted as distributions of points in a multidimensional Cartesian space. The genotype (CPPN) can be then queried for each coordinate of the space and give the phenotype representation of the genotype in multiple resolutions. In the same fashion, images can be constructed using CPPNs, where pixel coordinates are queried to the network and the grayscale-RGB values can be taken by the outputs of these networks. Figure 2.5, illustrates images encoded by CPPNs. Comparing the results with figure 2.2, is now understandable why this kind of encoding can capture in greater extent this problem domains, where symmetry is important.

Figure 2.4, illustrates how the mapping between the genotype and phenotype is done using generative encoding like CPPNs. A major asset of CPPNs is that they can generalize in all kind of resolutions. Considering the previous figure again, the CPPN is queried for all x, y coordinates of the phenotype two dimensional Cartesian space. The step of x, y sampling can be determined by the problem, since the inputs of the CPPN are the normalized coordinates $x, y \in [-1, 1]$. Thus, genotypes using this kind of generative encoding can be mapped in every resolution, making it straightforward to generalize into larger spaces. As the space of the phenotype becomes larger, a CPPN encoding a solution into this space which has a fixed topology (structure of network), is not affected

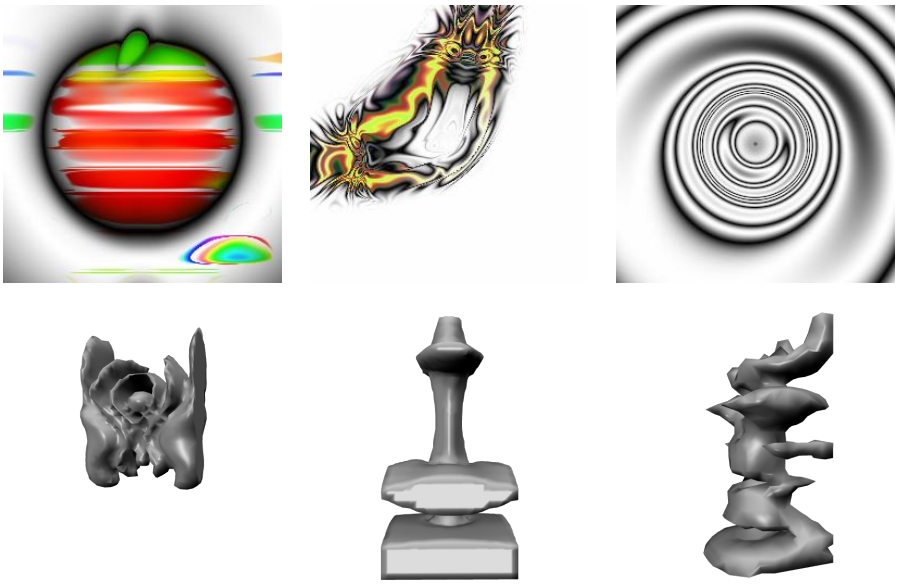


FIGURE 2.5: Compositional pattern producing networks (top) can encode truly complex structures and shapes (bottom) in the phenotype level. Source [7].

by the increasingly dimensions of the problem, a constraint that affects direct encoding schemes.

Compositional pattern producing networks have been used in many applications where symmetry and repetition can produce artistic two or three dimensional structures[8], and drawings [7, 9]. As these applications require more symmetrical properties than others, more inputs than Cartesian space coordinates are fed into the networks' inputs to achieve these aesthetic results. Some example input that can be fed into the network as inputs are the distance from the center of the space, or the distance from the center in only an axis.

2.4 Neuroevolution

Neuroevolution is an optimization technique using evolutionary methods as described in Section 2.2, where artificial neural networks take the place of simpler representations from the genotype (bit-streams) to the phenotype (solution space). ANNs can compute arbitrarily complex functions, learn and perform under the presence of noisy inputs and generalize to unseen sensory information. Neuroevolution requires only a measure of a network's performance at a task, which can be used as reward. The more complicated forms of chromosome representations can develop more complex "brains" for robot controllers. After each run, the sensory input of the task domain is given at the artificial neural network's input neurons, and the solution is given by the network's outputs, where the fitness of the specific "brain" can be evaluated. A major issue is the selection

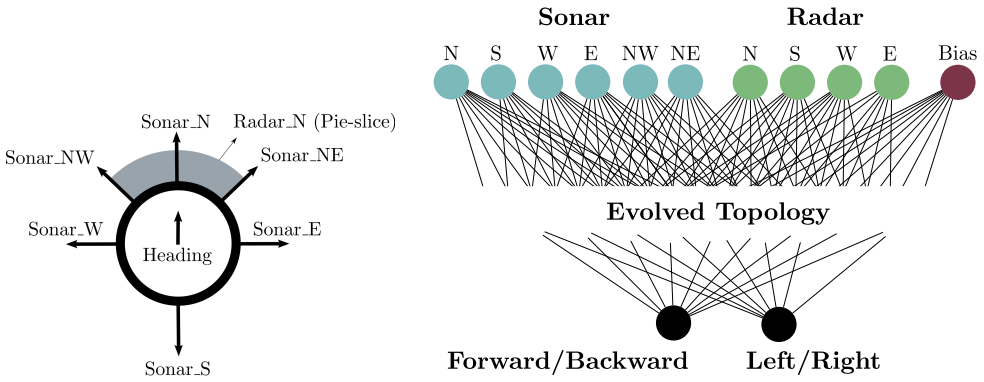


FIGURE 2.6: Robot controllers can be evolved through neuroevolution, where robot sensors are the input of neural networks, the output of the network directly control the robot.

of the network's *topology*, topology is the arrangement of the network's elements such as links and nodes, which represents the structure and how the information flows within the network. In early neuroevolution implementations the topology of the networks used, was fixed, meaning that the only elements of the networks evolving were the weights of the connections between the nodes.

2.4.1 Neuroevolution of Augmented Topologies

Neuroevolution of augmented topologies (NEAT) as it first introduced by [10] is a method of evolving neural networks, which evolves the topologies of the networks alongside its weights.

Originally, neuroevolution methods (methods that evolve ANNs), developed to capture difficult sequential decision as well as control problems, where sensory information is the input of these neural networks and decision are the outputs. NEAT is another method for evolving ANNs, whereas a few simple features are added, making it able to find solutions in more demanding problems. NEAT starts the evolution process with a population of networks with simple topologies. Through the generations instead of just fixing the weights of the networks' connections, topologies will become more complex allowing nodes and links to be added along the generations. Meaning that during evolution, more complex networks will be produced, this *complexifying* technique leads to capture more demanding problems, as it offers enough freedom to the evolution.

Figure 2.6, illustrates how sensor information propagates through a neural network and generates the motion control of a robot, which tries to drive itself close to a target position in a maze. All the sensor information (6-sonar inputs that output the distance from the closest obstacle in that directions and 4 pie-slice radar sensors, which are only activated when the target position is located within the angle of each).

Several aspects of this method are worth mentioning, with *speciation* be the most important one. Speciation is the procedure that protects new *species* until the have enough time to evolve, before comparing them with the rest of the population. For two individual genotypes (ANNs) to belong to the same species, their network topology must be similar, meaning that a threshold is set, and a function determines the numeric value of two network topologies' similarity. The age of each species protects them for competing in equal terms with more optimized species, giving them in this way time to evolve further towards the objective function.

2.5 CPPN-NEAT

Compositional pattern producing networks (CPPN) are identical to ANNs in respect to their structure, also can make use of the *complexifying* property, capturing in this way regularities in more complex problems. NEAT method can likewise evolve CPPNs in the place of ANNs, since it only needs minor modifications.

The resulted method that evolves this generative type of genomes (CPPNs) is called CPPN-NEAT [6], and its only difference with the original NEAT algorithm is in the way new nodes are added to the networks. The original NEAT algorithm uses to evolve ANNs which are using sigmoid functions to every node, so every new-added node will carry this function. In contrast, CPPNs use a variety of functions from a canonical set, so CPPN-NEAT assigns a random function from this set to every newly added node.

Experiments [6] have shown that the discussing method can indeed evolve CPPNs, capturing like this solutions in problems with geometrical nature, holding alongside essential properties of natural evolution. Furthermore, neuroevolution of CPPNs can also determine the connectivity patterns of neural networks in NEAT method [11].

2.6 Novelty Search

Traditional search within the framework of evolutionary algorithms needs an objective function, a function that leads the search towards "good" areas of the solution space, following the gradient of the fitness. Defining the fitness function is a straightforward problem most of the times, in a problem when a robot tries to get from its initial position to a target position in a room with no obstacles in between, a fitness function could be determined as the Euclidean distance between the final position of the robot and the target point, the closer it gets to the target the more points (higher fitness) the specific controller is rewarded.

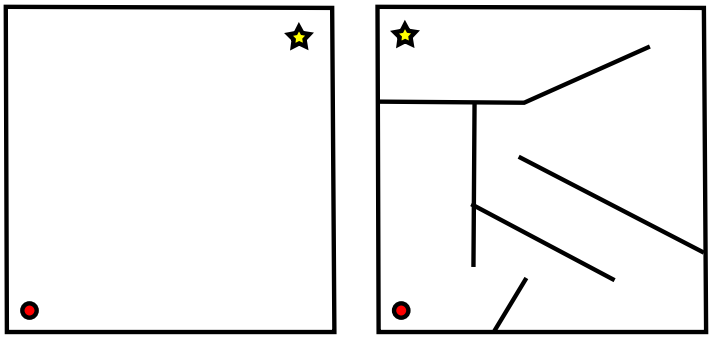


FIGURE 2.7: Objective functions can be devious. Maze example from [12].

When the objective function misleads the search

Such a measure as the objective function is greedy, driving the search directly towards more rewarding areas of the search can lead in many problems to local optima, misleading the search, which eventually stacks in these localities of the problem.

Considering the robot-maze example presented in [12, 13], a robot (red dot) is placed in a maze (fig. 2.7), the robot (fig. 2.6) has sensory information which are inputs to its controller ("brain"), the controller is driving the robot through the maze having only sonar sensor information, when its ultimate goal is to drive the robot to the target position (star) in a fixed time span. Naturally, to select a fitness function that can give enough information about how good a controller is, the euclidean distance to the target from the final position of the robot in the end of the simulation time is measured, providing knowledge to the search. For the first maze (left) when no obstacles are between the robot and its target, the objective function is reliable, since the euclidean distance to the target indeed informs the robot how close it is located. In the second maze (right), using the same fitness function, search can be misled. In this example maze, achieving high fitness does not mean that the robot is actually close to the target. Driving north in this maze following the increasing fitness leads to a wall that cannot be passed by the robot, therefore, exploration is needed in low-fitness areas which will allow the robot to reach the target point with the maximum fitness. The deceptive nature of the fitness function in this problem can be found in a lot of optimization problems, and the walls in this maze clearly denote problems where this fitness landscape can be found.

Natural evolution is not a search for fitness

Using an objective function in evolutionary computation and typically reward individuals which are closer to an objective, is far away from natural selection in the evolution

process, where exploration is allowed as long as, the criteria for survival hold [13]. Driving search towards promising parts of the fitness space, whereas local optima may be present, ensures that other areas of the search will not be explored, leading the search to stay and explore the nearby area, while more promising regions are far away in solution space. Solutions located in these regions are called *stepping stones* [12–15] and are points in the search space that may not be good, as far as their objective values is concerned, but can eventually lead to better or a global optima.

Search for Novelty

Novelty search [12–15] unlike traditional fitness-based search is an alternative way of optimization towards an objective function without having knowledge of this objective. In simple words, it is looking for a solution to a problem without knowing how close it is to actually solve it, which turns out to be a major impact to the increased performance of this method in several domain problems. A similar concept can also be found in reinforcement learning when the need for exploration has to be preserved through the learning process.

What novelty search seeks for, is how interesting is a new solution in respect to all previously found ones. To define "interesting" we need to move our point of interest into behavioral space, which is a function of each phenotype just like the fitness. Nevertheless, it fully describes the behavior without implying directly the fitness function. As an example someone can think of, the final position of the navigation robot, or the trajectory of it, in the previous robot-maze example. Rewarding behaviors of the phenotype that are different from the previously found, evolution is being driven to visit new points in the behavior search space.

One significant point here is that the behavioral space in some domains can be limitless. However, a valid behavioral metric could be found, excluding behaviors that are meaningless or do not comply with the natural limits of the problem. Additionally, enumerating all possible solutions in the behavior space is similar to a brute force approach which could have the same results. On the other hand, the genotype's search space can be also infinite, especially in neuroevolution methods like NEAT, in where ANNs can grow over evolution time. A bounded space of understandable behaviors is then the key idea of novelty search, whereas increasingly complex behaviors present to the evolution as the complexity of the genotype grows along.

Multi-objective optimization can also make use of novelty metric trying to optimize both objectives fitness and novelty at the same time [16]. Another method that exploits the

diversity of the produced genomes in order to map the phenotype to the fitness is also proposed by the literature [17].

Is novelty search the similar to random?

Initial thoughts are converging that novelty search is completely random search, constantly looking for something new in the vast space of behaviors is similar to evolving random robot controllers without caring about behavioral aspect of their phenotypes, hoping that enough exploration will be done in both genotype and phenotype spaces. Thus, having no information about the actual behaviors the evolved phenotypes produce, can harm the evolution as different and more complex genotypes can easily produce similar behaviors. The novelty in behavior level assures that the search will explore deeply the behavioral space with the hope that a *fit* behavior will be found. Aside from that, novelty search does not perform backtracking which ensures that it will constantly drifts away from already found behaviors, in the same time, there is no such guarantee in random search. Therefore, it is certain, and proven later in this thesis, that no exploration in the behavior space will be performed by random search.

How novelty can be measured

As fitness is a function to measure the "goodness" of an individual, novelty measures how different is an individual against all previous found by the evolution. To define different, novelty metric measures the difference in the behavioral space of the phenotype. Given the phenotype's behavior x, a novelty measurement could be a function of x, $f(x)$, which computes how different (novel) is the specific behavior in respect to a set of other behaviors S in behavior space. As defined in [12, 14], *sparseness* can give a good measurement of how sparse is the area of a newly introduced behavior. Given the behavior, we can compute the sparseness by:

$$f(x) = \frac{1}{k}\sum_{i=1}^{k} dist(x, S_i) \tag{2.1}$$

, where S is a sorted set of the closest behaviors, so that, the average distance from the k-closest behaviors.

Algorithm

There is no need of extensive modification in any evolutionary algorithm in order novelty search to be implemented, apart from replacing fitness with a novelty measure. To push

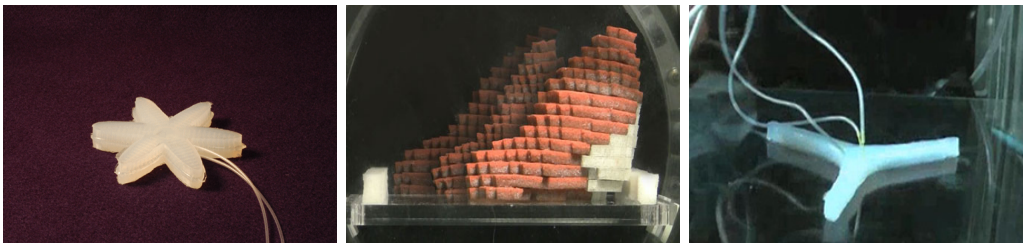


FIGURE 2.8: Soft robots can be actuated through air pressure tubes (left), pressure variation (middle), even internal explosions (right).

search to visit new areas in the behavior space we need to reward every novel behavior coming up during the evolution. For this reason, storing novel reference points in space (behaviors) during the evolution is inevitable. The sparseness of a new behavior is computed by equation 2.1, resulting in a numerical value that implies how novel is the found behavior. If the new behavior has a novelty value more than this threshold it is also stored in the novel individuals' population. Apart from comparing any new behavior with all the novel stored ones, the newly produced can also be confronted with the entire set of behaviors produced in the same generation of the evolution.

2.7 Soft Robotics

Soft robotics is a field of research dedicated to the science and engineering of soft materials in mobile machines, even though it is on an early stage it seems really promising. As the name suggests soft robots [18, 19] are made completely of soft materials mimicking animals or animal-parts that consist of soft tissue (elephant trunk, tongue, worm, octapus, etc.). Having no rigid parts, the degrees of freedom can explode and the possible ways of motion can become extremely complicated. In traditional robotics, joints and rigid parts predefine the set of possible movement space and sometimes restrict the robot's locomotion strategy or *gait* to a specific set. In soft robotics, the absence of rigid parts can on the one hand make the design of the locomotion strategy exceptionally tortuous, on the other hand though, the gait alternatives are limitless.

The design and development of soft robotics is not an easy task, soft grippers can be developed, made throughout by soft materials, however the actuation of such a soft structure is the most challenging task. Actuating soft materials can be done in many ways including pneumatic systems [20, 21], hydraylic, internal body explosions, pressure tubes, temperature changes and others [22, 23]. Figure 2.8 illustrates three different ways that soft-robot bodies can be actuated. Gripping mechanisms [24] can softly and gently conform to objects of any shape and hold them with uniform pressure. This gripping function is realized by means of a mechanism consisting of links and series of pulleys

FIGURE 2.9: Autonomously actuated soft robot [26].

which can be simply actuated by wires. Regardless traditional ways of actuating soft material robots, three dimensional printing is now giving the freedom for multi-material structures to be created, which also explodes the number of possibilities for the design of a soft structure such as a gripper for instance. Topological optimization techniques can be applied [25] for producing functionalities in the design. Autonomously actuated soft robots [26] (fig. 2.9) can also be designed, having multiple advantages over rigid body robots, such as resistance under extreme temperatures, locomotion on terrains of variant types.

It may seem on early stage, nevertheless, soft robotics research field is growing fast. Some of their characteristics make them interesting to explore, such as the infinite number of degrees of freedom, the variety of materials (mostly elastic) that can be used in the contrary to rigid robotics that are mostly made out of metals and plastic. Nevertheless, structure design and control of soft robotics remain challenging mostly because of their soft bodies can only be represented in continuous state space, where only analytical methods can be proven successful.

To sum up, a lot of research is being done in the field of soft robotics and materials, locomotion capabilities of soft robots as well as the fact that can passively move makes them an interesting topic for present and future research. Finally, considering also that soft materials can be more friendly than conventional robot materials to humans, human-robot interaction can become in this way safer [27].

2.7.1 Soft Robotics in Simulation

Most work to simulate interactions and deformations within and between soft material bodies are mostly focused on the graphical part of the problem, sacrificing the accuracy of

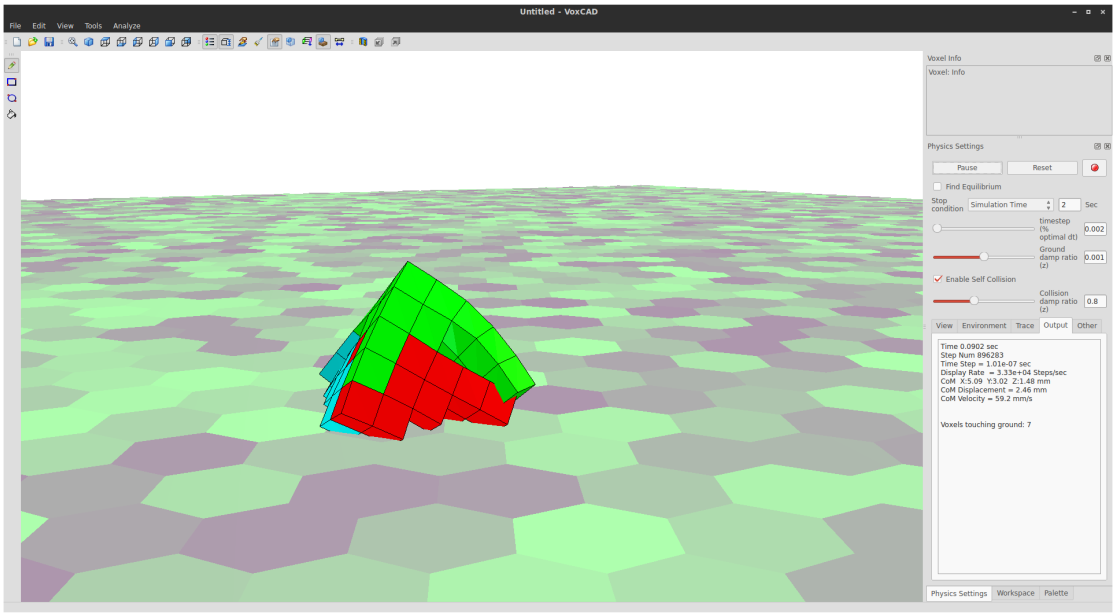


FIGURE 2.10: VoxCAD (Voxel CAD),a cross-platform open source voxel modeling and analyzing software.

the simulation. Three dimensional meshes can represent these bodies including dynamics of their materials.

A recent work though, *VoxCad* simulator [28], is focusing mostly on the engineering side of the soft material interactions, not at the expense of a low frame rate. *VoxCad* is a modeling and analyzing open-source software that can simulate soft material deformations and interactions. In figure 2.10, VoxCad software is illustrated during the simulation of the soft body robot depicted in the simulator.

VoxCad cannot model and simulate three dimensional meshes, yet a lattice is used to represent the 3D workspace where voxels (three dimensional pixels) can be assigned different materials. Materials have properties such as the elasticity of the material, density, Poisson's ratio, coefficient of thermal expansion (which determines how materials will be expanded in respect to the environment's temperature), temporal phase in respect to the temperature period, and the friction coefficients to the ground. Materials can also be mixed together to create a new type of material.

Materials themselves are passive and cannot actuate without external trigger. In this simulator this external force that can actuate the materials is the temperature of the environment. The main variables of the environment is the base, the amplitude and finally, the period of the temperature. Furthermore, the gravity acceleration of the environment can vary.

Throughout this thesis, a structure or a soft robot will refer to a set of connected voxels (not unconnected parts) within the lattice space. For universal or experimental settings used during the simulations you can see Appendix A.

Chapter 3

Related Work

Evolving virtual creatures in a simulation environment first presented 20 years ago by Karl Sims. In this chapter presents valuable related research work in the areas of evolution in virtual but also physical environments. Novelty search has been used in past work to evolve virtual creatures with no success.

3.1 Evolution of Virtual-Physical Robots

Robot controllers can be evolved through evolutionary algorithms on simulated (virtual) robots, although evolutionary methods can be applied to physical robots [29] as well, when no damage can occur due to exploration of the action space.

Novel systems, that make use of evolutionary methods to evolve artificial neural networks which can control the morphology of rigid body parts connected with joints, and forces applied to the joint so virtual creatures (fig. 3.1) can be produced [30] in a physical three dimensional world. Different fitness measures also give the possibility of the evolution of diverse creatures in respect to these measures. This genetic encoding defines a hyperspace of infinite number of possible creatures and behaviors, and when it is searched using optimization techniques like EA, a variety of successful and interesting locomotion strategies emerge, some of which would be difficult to invent or build by human designers.

Evolvable virtual creatures play an important role in computer graphics when the need for natural looking morphologies can save time from designers. While previous work [31] ended up with unnatural looking shapes and behaviors due to vast solution space. A system that uses Lindenmayer systems [32] (L-systems) as the encoding of an EA for creating virtual creatures. Creatures evolved by this system have hundreds of parts, and

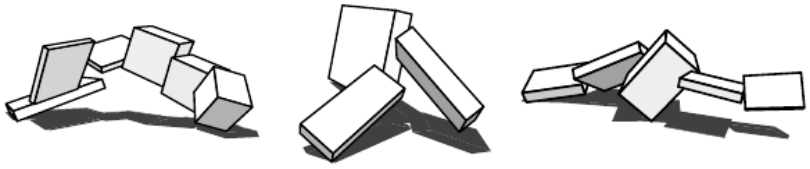


FIGURE 3.1: Karl Sims, "evolution of virtual creatures" [30].

the use of an L-system as the encoding resulted in creature morphologies that have a more natural look.

Evolutionary robotics have shown the ability to evolve complex designs which can complete tasks in their native environment are trained for. However, these complex designs do not serve well engineering purposes of EA, whereas these complicated designs are hard to applied or impossible to made in a physical robot. Generative representation used in [33], accomplishes to turn the representation into a construction plan which uses simple robot components in a regular way, moving in this way more effective in the design space. As direct encoding schemes have trouble capturing geometrical properties of the problem, generative encodings like CPPNs can be used in order to take advantage of a problem's regularities and symmetries.

HyperNEAT [11], which is a method to evolve CPPNs which will determine the topology and the weights of ANNs, shows promising results in evolving the gaits of legged robots [34], where direct encodings have trouble. Natural evolution is the only process which instead that evolving only the brain of biological organisms, it also evolves the morphologies of them. CPPN-NEAT [6] can be used as a generative encoding EA which can evolve both features of virtual robots [35, 36], implying that more complex creatures than designers imagination can be created in such a setting. It is also possible that a lower resolution is used at the first runs of the evolution to save computational time without significantly degrading the quality of evolved structures and later a higher resolution for the details optimization.

It has also been shown that evolving objects with encodings based on concepts from biological development like CPPNs can be a powerful way to evolve complex, interesting objects [37] , which should be of use in fields as diverse as art, engineering, and biology. There is enough evidence as well, that information provided to the CPPNs can also bias the outputs of these networks.

Apart from the use in robot-bodies design evolution, EA techniques coupled with indirect coding schemes allow the evolution of the morphology and the motion control of soft bodies, in this case multicellular animats [38] in a 2-dimensional fluid-like environment. Both the developmental program and motion control are encoded indirectly in a single linear genome, where a genetic algorithm can be applied to evolve it.

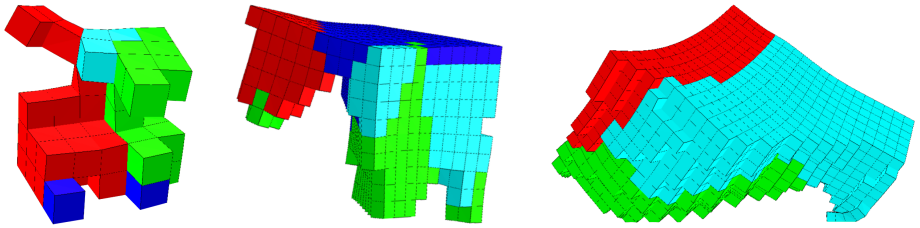


FIGURE 3.2: Evolution of soft robots' morphology by indirect encoding (CPPN) [40].

With the excel of 3D printing, soft multi-material robot bodies can actually be produced using simple material types. These soft structures made only by soft materials can be simulated [28] allowing for the evolution of their design without the costs of production. As it is first shown in [39], the automated design of three dimensional bodies which can obtain functionalities through the distribution of hard and soft materials inside the three dimensional space. The robots were evolved (EA) and tested for a single-direction locomotion displacement, whereas it has also been proved, testing a soft material robot inside a pressure-chamber that, the actual error compared to the virtual one in the simulation was small.

Evolution of soft material robots as was shown in [39] can produce structures able to locomote. The possibility of evolving these soft structures using indirect coding was of interest to be exploited by [40]. A powerful generative encoding, CPPN [6], used to generate soft voxel-formed three dimensional structures (fig. 3.2) like in [39], coupled with the use of NEAT algorithm which ensures the increase of complexity of the networks produced, were purposed for evolving these soft-robots. The superiority of this kind of generative encoding was verified, showing how CPPNs can take advantage of the geometrical properties they show. Evaluation was done by a simple displacement measure. Yet, evolution tended to stick to different kinds of locomotion strategies and morphologies as the fitness function was penalized for different kinds of parameters. Furthermore, it has been shown that evolving morphologies (CPPNs) in lower resolutions and then applying the same networks for higher resolution structures can be beneficial, since the locomotion behaviors in lowers structures apply also in higher, saving in this way computational time.

3.2 Evolving Virtual Creatures by Novelty Search

In problem with such high dimensionality as evolving both the morphology and locomotion strategy of artificial creature in simulated or physical environments, evolution does not explore the solution space enough, sticking only with first easiest to exploit morphologies. However, novelty search, a technique that explicitly rewards diverging,

can potentially mitigate such convergence. Methods for evolving such virtual creatures like in [30], can utilize novelty search [41], and be far more explorative in the search space. Behavior novelty defined as a measure between morphological properties of the produced creatures driving the evolution to explore more diverse morphologies. This kind of defined behavior cannot led the larger diversity of creatures to move, as different produced morphologies does not guarantee that some of them will actually move. However, combining fitness and novelty metrics through local competition led to improved results, whereas novelty search alone failed.

Chapter 4

Method

In the previous chapter (Ch. 3), related work in the domain of evolutionary robotics in virtual simulation worlds were presented. Pure novelty search in evolving three dimensional showed that it cannot compete with fitness-based search especially in the case the behavior defined does not contain any fitness information. It is of interest to show how novelty search can compete fitness-based search in evolution of morphology of soft-body robots in a virtual simulation. In this chapter, an introduction to the problem specifications this thesis is concerns about, as well as, a comprehensive documentation describing the methods used, is given.

4.1 Problem Introduction

Recent work in evolutionary robotics shows that evolving the morphologies of soft robotics is possible through compositional pattern producing networks coupled with NEAT evolutionary algorithm. VoxCad simulator provides a test-bench for analyzing soft robot bodies that can be actuated through environmental changes, in this case the temperature. In addition to that, recent work by [40], shows that very interesting morphologies can be evolved by the CPPN-NEAT algorithm in the specific soft-robot simulation environment.

VoxCad

As far as the simulation settings are concerned, it is not of interest for this thesis to explore the best not only environmental but also material settings for the evolved soft-robots. For the simulation of the soft material bodies, VoxCad's underlying physics engine *Voxelyze* was used as a stand-alone software to analyze the soft structures without

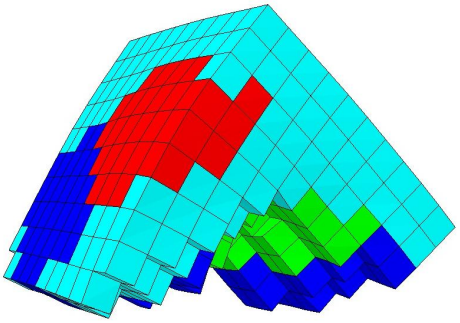


FIGURE 4.1: Soft robot uses four materials (two active, two passive), morphology evolved penalizing actuated materials.

rendering. Table A.1, describes and presents the values used in different variables of the simulation.

Materials

Figure 4.1, illustrates a soft robot consisting of all four materials used in the experiments. Red and Green are the only actuated materials with non-zero and opposite thermal expansion coefficients, meaning that their phase in respect to the actuation from temperature changes is equal to half a circle, green voxels contract the same time red expand and vice versa, mimicking living organisms' muscle tissue. The two additional materials represent soft non-actuated tissue that can be soft (soft tissue) or hard (bones). Cyan voxels are soft, having five times smaller elastic modulus of their material than Blue which have 50 `MPa`.

4.2 Direct-Generative Random Soft-Robots

To evaluate all the following evolutionary methods used, information about the performance of random generated morphologies must be present. In order to achieve that, two random approaches which will also help the understanding between direct and indirect coding implemented. Direct coding which is a straightforward encoding scheme assigns randomly the presence of a voxel in a lattice's coordinate, if a voxel will be created then it will be assigned a random material from the palette. The probability of adding a voxel is 0.5, after all voxels have been assigned a material, unconnected parts of of the structure will be removed, keeping only the largest connected structure in the lattice.

The indirect way of producing random morphologies, follows a different method of assigning materials to voxels, having a set of rules that are followed in order a morphology

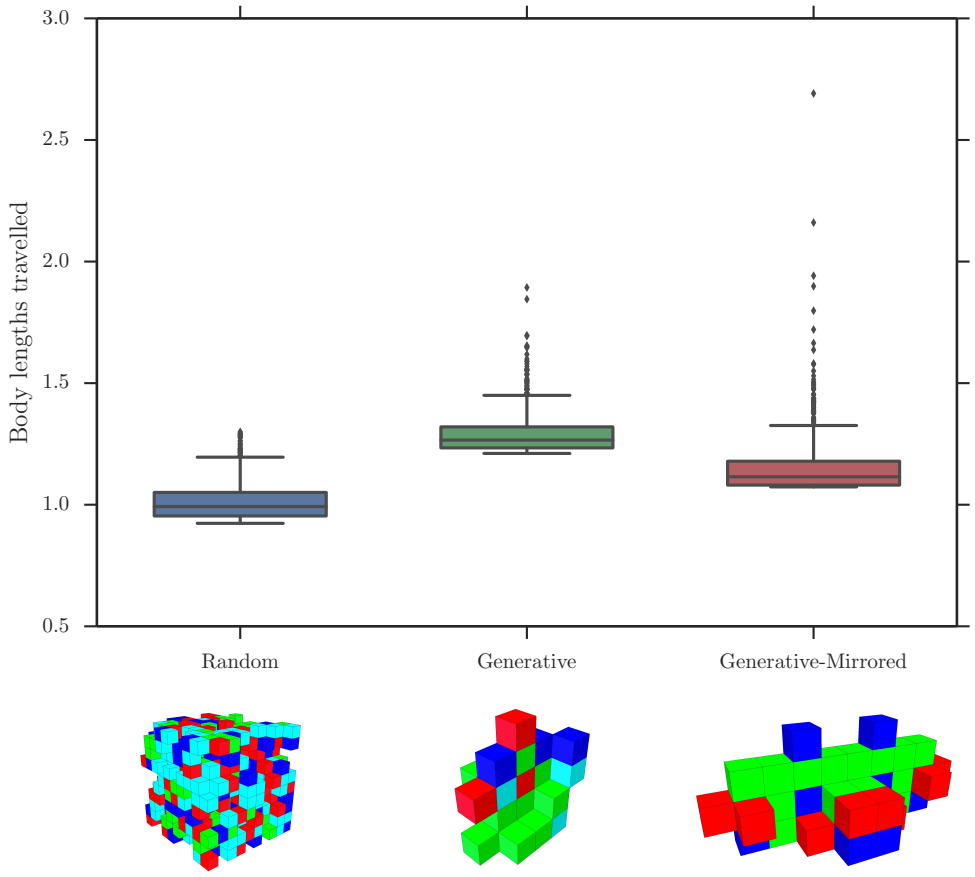


FIGURE 4.2: Generative encoding creates more natural morphologies even in random schemes.

to be produced. This methods holds two probabilities, the one refers to the probability of adding a new voxel in the already made structure, the next one denotes the probability the material of the new inserted voxel will be the same as its connection's material. First, a voxel of random material is inserted in a random coordinate into the lattice. If a new voxel is to be added a connection (voxel) is chosen from the already inserted ones. The side of the connection is chosen from a uniform distribution out of all valid sides. In this generative process there is also the possibility of creating structures in half of the lattice space, and then mirror the soft structures in both halves, creating in this way symmetrical morphologies.

Considering these two methods, the difference between direct and indirect coding is becoming easier interpreted. In the direct process, a probability determines the presence and the material of every voxel in the lattice. On the other hand, in the generative method a set of rules and probabilities define the structure that is going to be produced into the available space.

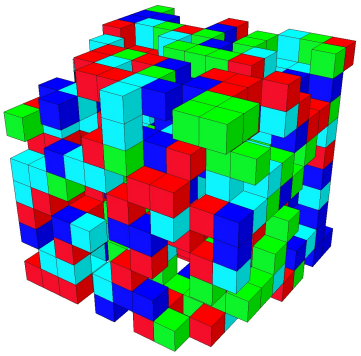


FIGURE 4.3: Direct encoding cannot capture the geometrical properties of some problems.

Figure 4.2, illustrates not only the actual performance (top-1000 soft-robots from 30000 total runs for each method) of the previously described method but also one of the best performing robots of each method. Both generative methods outperform the direct one, mostly because there is no structure in the created creatures, another one reason is that since there are no rules in the construction of the random structures resulting in morphologies with huge number of voxels, something that makes them difficult to locomote. On the other hand, generative random methods create more compact structures which can move easier due to their size and some of their geometrical features. For the mirrored approach even though the average performance is slightly worse than the plain method it actually performs way better in some distinct cases. Inserting some geometrical properties like properties resulted in getting more successful locomotive structures. All methods achieve an average displacement of the soft-robots around or more than 1 body length, which considered to be very low in respect to the robots generated by the evolutionary methods.

4.3 Direct-Encoded Evolutionary Soft-Robots

In the previous section, indirect and direct random morphology methods were implemented and tested, failing to produce any decent locomotion gaits for the soft structures. Considering how vast is the solution space, random approaches are doomed to fail in a definite number of tries.

Hence, a more sophisticated method is tested here. Direct encoding genomes coupled with a simple genetic algorithm is a successful approach in evolving robot controllers. As it was previously stated in ch. 2, mutations and crossovers of real value streams, search the space effectively succeeding in difficult optimization problems. GAlib C++ library [42], used for the implementation of this method.

Representation of the genotype

As in every direct encoding scheme, genotype can be represented by a stream of bits, which length is equal with the number of the dimensions of the problem. In this case, using 4 materials but also trying to encode the presence of each voxel in the lattice as another dimension of the problem. Analytically, its length can be represented by a stream of length equal with the number of voxels in the lattice times the materials used plus one denoting the presence, described by the following equation:

$$|Genome| = (l_x \times l_y \times l_z) \times (1 + |p|) \tag{4.1}$$

, where l_x, l_y, l_z are the dimensions of the lattice space, and p is the palette of materials.

$$Genome = \underbrace{01010\ldots011011}_{\text{Presence}} \ \underbrace{10101\ldots110011}_{Material_1} \ \ldots \ \underbrace{00011\ldots111110}_{Material_n}$$

The above stream of bit illustrates how a soft-structure in VoxCad environment can be represented. Each of the values of the stream is represented by a value between zero and one, which represented in the algorithm by a stream of bits. The mapping from the genotype level to the phenotype is straightforward in this case, the first stream of values is used to determine the presence or not for a voxel, while in case of a presence the other n streams are used and the maximum value in specific positions of the streams determine the material is going to be used.

Considering the representation of the genome, as well as the geometrical nature of the problem it self, it is valid to say that we do not expect that direct encoding will capture this major property of the problem (Fig. 4.3).

4.4 Generative-Encoded Evolutionary Soft-Robots

Direct encoding schemes lack the regularity of their designs in the phenotype level leading in morphologies that cannot produce any coordinated locomotion behaviors. Generative-Indirect encoding (CPPN) as it previously explained in detail, serves this function. Producing regularities in the phenotype space, and capturing geometrical properties of the optimization problem, it is expected to produce fine locomotion strategies and morphologies of the soft structures [40].

Compositional pattern producing networks are built up by a set of canonical functions which force the outputs of the network to produce repetitive, symmetrical and geometrically interesting patterns. Since, CPPNs must queried for every coordinate of the lattice space, the input nodes of the CPPN were assigned to x,y,z normalized coordinates

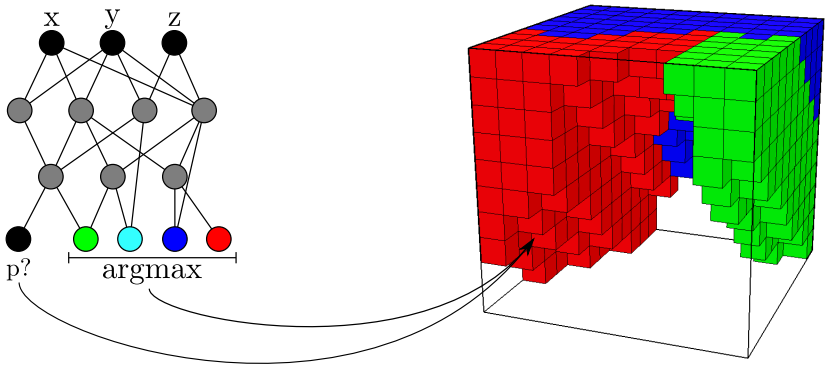


FIGURE 4.4: Each genotype is queried for every coordinate inside the lattice, its outputs determine the presence of a voxel and the type of its material.

in $[-1, 1]$, following [40]. It should be pointed out here that more inputs could be added to the CPPN-network, such as the the distance from the center point of the Cartesian space (simulation lattice) as described in [6], which naturally adds more bias towards symmetrical structures, than CPPNs already do. What is more, creating perfectly symmetrical structures is not much of an interest to show, since CPPNs, as shown later in this thesis, can evolve symmetrical shapes without this symmetrical extra information input node. Figure 4.4, illustrates the topology of a random CPPN network with the input and output nodes described above. Every connection node between input and output nodes is called the network's *topology* which can be variant and be evolved alongside the weights of the connections in an evolutionary method like NEAT. The part of the network that is between the input and output nodes, is the part of the network that is described by the genotype and the one is going to be evolved and altered during the evolution. Once again, the presence or not of a voxel in a lattice coordinate is determined by a single output of the CPPN called p and the selection of the material out of the palette is again determined by n outputs, the node will the maximum value in the output will determine which of the n materials will be used.

4.4.1 How CPPNs can be evolved?

The evolution of these indirect representations of the phenotypes can be evolved with any method able to evolve artificial neural networks, since they are identical with CPPNs. CPPN-NEAT (first described in Ch. 2), is a method of evolving CPPNs with NEAT evolutionary method, which was chosen to evolve those networks as it has proven successful in previous work in the same setting [40]. HyperNEAT `v4.0 C++` by J. Gauci code (url: `https://github.com/MisterTea/HyperNEAT`) was used for the experimental implementation. Algorithm 4.1, presents the pseudocode for the evolution under CPPN-NEAT method. In addition, a brief explanation of each function used in the algorithm follows:

Algorithm 4.1 CPPN-NEAT evolution

```
1: population = ∅
2: species = ∅
3: generation[0] = initial_population()
4: for i = 0 to max_generation do
5:    species = species ∪ speciation(generation[i])
6:    evaluation(generation[i])
7:    adjust_fitness(generation[i], species)
8:    selection(generation[i], species)
9:    generation[i + 1] = reproduction(generation[i])
10:   population = population ∪ generation[i + 1]
11: end for
```

Initial population Before the evolution starts, an initial population must be produced, random genomes (CPPNs) fill up the population.

Speciation ,takes place and split the population in different species or adds individuals to already existing species in respect to their network's topologies. However, all firstly introduced genomes belong to the same species, due to the identical topology of their CPPNs.

Evaluation Once the population is filled with new individuals, these have to be evaluated. Simulation is take place for each of the individuals of the population, whereas each one of them is awarded with a fitness value.

Fitness adjustment After all individual are evaluated, each species is assigned a value which is the sum of the fitness values of the individuals belonging to the species divided by the number of the individuals. The way it is been decided how many individuals each of the species will breed is directly determined by the average fitness of each species.

Selection As soon as the number of new individual each species is determined, only the top 20% of the species population will reproduce, the rest population will "die".

Reproduction There are three ways, for the selected individual inside each species to reproduce. The first is called *elitism*, meaning that the best of each species will copy itself in the next generation. The next two are *mutation*, which changes the genome of one individual slightly and creates a new genome for the next generation, and *crossover* which is the most natural operator and it uses mixes two parent individuals to create a new genome.

Algorithm 4.2 CPPN-NEAT with novelty search

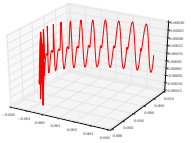
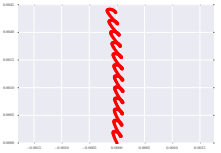
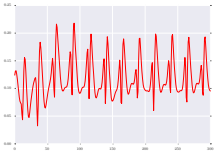
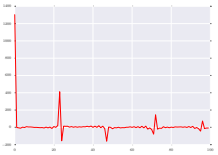
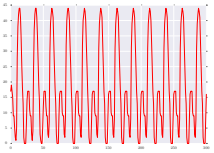
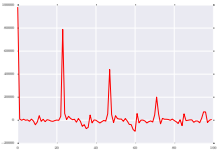
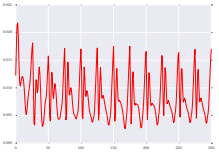
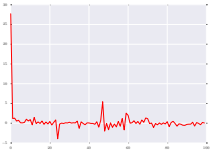
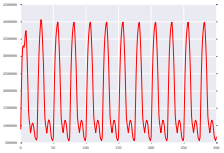
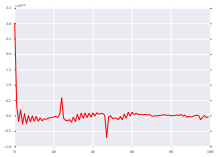
```
1: population = ∅
2: novel_inds = ∅
3: species = ∅
4: generation[0] = initial_population()
5: for i = 0 to max_generation do
6:   species = species ∪ speciation(generation[i])
7:   evaluation(generation[i])
8:   for all ind ∈ generation[i] do
9:     novelty = sparsity(ind, (generation[i] − ind) ∪ novel_inds)
10:    if (novelty ≥ novelty_threshold || novel_inds == ∅) then
11:      novel_inds = novel_inds ∪ ind
12:    end if
13:  end for
14:  adjust_novelty(generation[i], species)
15:  selection(generation[i], species)
16:  generation[i + 1] = reproduction(generation[i])
17:  population = population ∪ generation[i + 1]
18: end for
```

4.4.2 Novelty search

Novelty search as first presented in Ch. 2, requires small changes in an evolutionary algorithm. Fitness is replaced by a novelty metric which determines how different is a phenotype's behavior in respect to novel behaviors found before. Sparsity (eq. 2.1), is used to dermine this measure, whereas every individual is compared not only with the previous novel behaviors but also with the current generation individual behaviors.

The algorithmic adjustments are presented in Algorithm 4.2, where the pseudocode of novelty search is presented. Instead of just returning a fitness value, function **evaluation** has as an output the behavior of the individual evaluated, as well as its fitness for comparing purposes. Function sparsity, computes the sparseness (eq. 2.1) of a specific individual's behavior in the behavioral space, calculating the mean distance from the k-closest behaviors. Following the evaluation of each individual and the returned point in the behavior space, if the novelty metric is larger than a threshold, this individual will enter the novelty individuals' set. The fitness adjustment of the previous code example is becoming novelty adjustment following the same functionality, selection and reproduction methods are working in the same fashion, whereas they are comparing the novelty of the individual in the place of their fitness.

TABLE 4.1: Behaviors used for novelty metric computation, to evolve morphologies for the soft-robots.

Behavior	**Sampling**	**DFT**	**Example**	**Description**
3D-trajectory	1 KHz			Set of three dimensional sampled points of the robot's center of mass during simulation.
2D-trajectory	1 KHz			Set of two dimensional ground projection sampled points of the robot's center of mass during simulation.
Pace	1 KHz			Set of robot's pace sampled values.
DFT-Pace	100 KHz	✓		Set of the robot's pace sampled values transformed into the frequency space.
VTG	1 KHz			Voxels touching the ground in each sampling time.
DFT-VTG	100 KHz	✓		Voxels touching the ground transformed into the frequency space.
Pressure	1 KHz			Maximum pressure among the connected voxels.
DFT-Pressure	100 KHz	✓		Maximum pressure among the connections transformed into the frequency space.
KE	1 KHz			Maximum kinetic energy of voxels.
DFT-KE	100 KHz	✓		Maximum kinetic energy of voxels transformed into the frequency space.

4.4.2.1 Behavior in novelty search

In the interest of novelty to be defined, behaviors should be defined that make sense in our problem setting. On the one hand, trying to optimize locomotion strategies of soft-robots under variant environmental conditions in fitness-based methods, a good measure is the displacement under a limited time simulation. On the other hand, novelty search is in a need of a behavior metric that encodes these fitness attributes inside. Novelty search forces the evolution to try new behaviors, if the objective cannot be encoded in the behavior, the search for novelty will become random. Behaviors that describe the morphology of the evolved soft-robots have failed [41], since search is then forcing new types of morphologies without caring about the actual target of the evolution. As an example, the number of voxels a soft robot has, is not a well-founded behavior metric, since the search will reward new structures with different number of voxels from previous ones, there will be no exploration in some metric that affects the actual target of the evolution which is to produce and evolve good locomotion strategies. Table 4.1, presents all behaviors used for novelty metric computation, with the sampling rates of the recorded values during the simulation and a description.

All these behaviors designed having in mind that enough information about the locomotion success must be encoded into the behavior's recorded signal. Trajectories (2D and 3D), incorporate all the needed information, such as speed, displacement, locomotion strategy. To avert from same trajectories in all possible directions, trajectories are normalized, meaning that their starting coordinate in both cases (2D and 3D) is always the start of the axes, and the coordinates of the trajectory are rotated so their center of mass is normalized to meet a certain angle. Computing the difference of two trajectories, the Euclidean distances of all coordinates of the one trajectory are computed in respect to coordinates at the same sampling time.

Pace is also a very informative behavior metric as it directly measures the speed of the robot. Voxels touching the ground can also produce information about the locomotion strategy but not enough about the actual performance displacement-wise. Hopping robots that move fast can have same behaviors with hopping robots with zero speed. Maximum pressure among the voxels' connection is one more behavior metric, pressure is expected to become higher as structures move faster and interactions with the ground eventually getting harder. Finally, maximum kinetic energy is a different behavior that straightly determines the displacement of the voxels in the structure. For all behaviors but trajectories, the Fourier profile of their signal can also be used as a behavior signature of the individuals, which can also eliminate shifts of signals in time-axis. To compute the difference of over two signals a straightforward method is chosen. Subtracting the one signal from the other taking the absolute differences and sum them up to compute

one single value that describes how variant the two signals are. Furthermore, for the Fourier transformations of these signals, the first twenty coefficients are compared, the absolute sum difference again determines the difference.

As we have now defined, what is a behavior and how one behavior can be different from another one to be considered novel, behaviors that are considered to be novel are stored in a list helping in this way the evolution to avoid generating similar behaviors (Alg. 4.2).

Chapter 5

Results

Following the comprehensive analysis of the evolutionary methods used in the experiments in the previous chapter, in this chapter the performance of these methods is compared and analyzed. Pure novelty search is compared in respect to the fitness measure used in the simulations (displacement of soft-robots in body-lengths), against fitness search. Questions, as far as what happens in the average fitness and the champion fitness of each generation within novelty search, answered in the following sections. Additionally both search methods are compared in respect to the number of novel behavior they generate during an evolution experimental run. The effect the behavior metric selection has in novelty search is also considered, as well as, the number of closest behaviors in sparsity equation plays also an important role in the evolution towards diversity of behaviors. Selection techniques such as competition within species are also used in both search methods, to determine the effects they have in the performance towards the specific goal it has been set. A new method is proposed for incorporating fitness information into novelty search without unbalancing the search for novelty and its properties. Last, the performance of both methods are now investigated within variant gravity levels, in order to show that gravity conditions do not have an effect towards a specific search method, as well as to examine different evolved locomotion strategies under different gravity conditions.

As in [40] and for comparison purposes, the population of each generation used is 30, and the number of generations of the evolution is 1000. For more details about the evolutionary algorithm and simulation settings used, see appendix A. Due to computationally expensive simulations, lattice sizes less than 10^3 have been used as well, more specifically experiments have been done under $5^3, 7^3, 10^3$ lattice space.

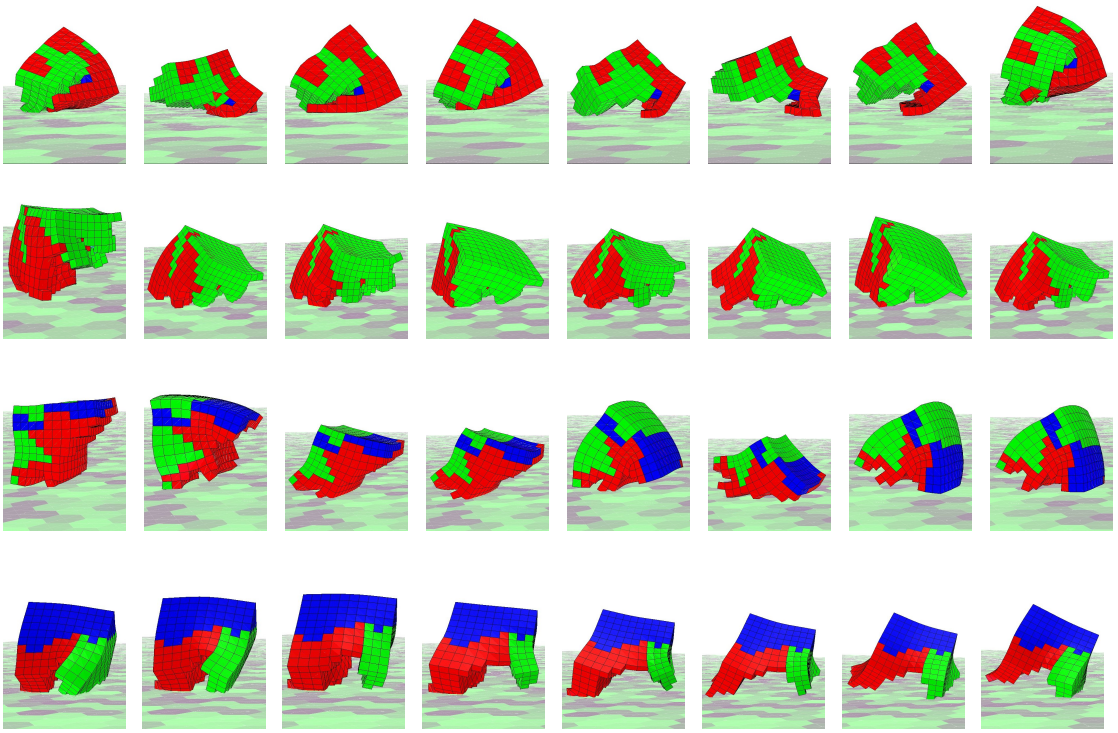
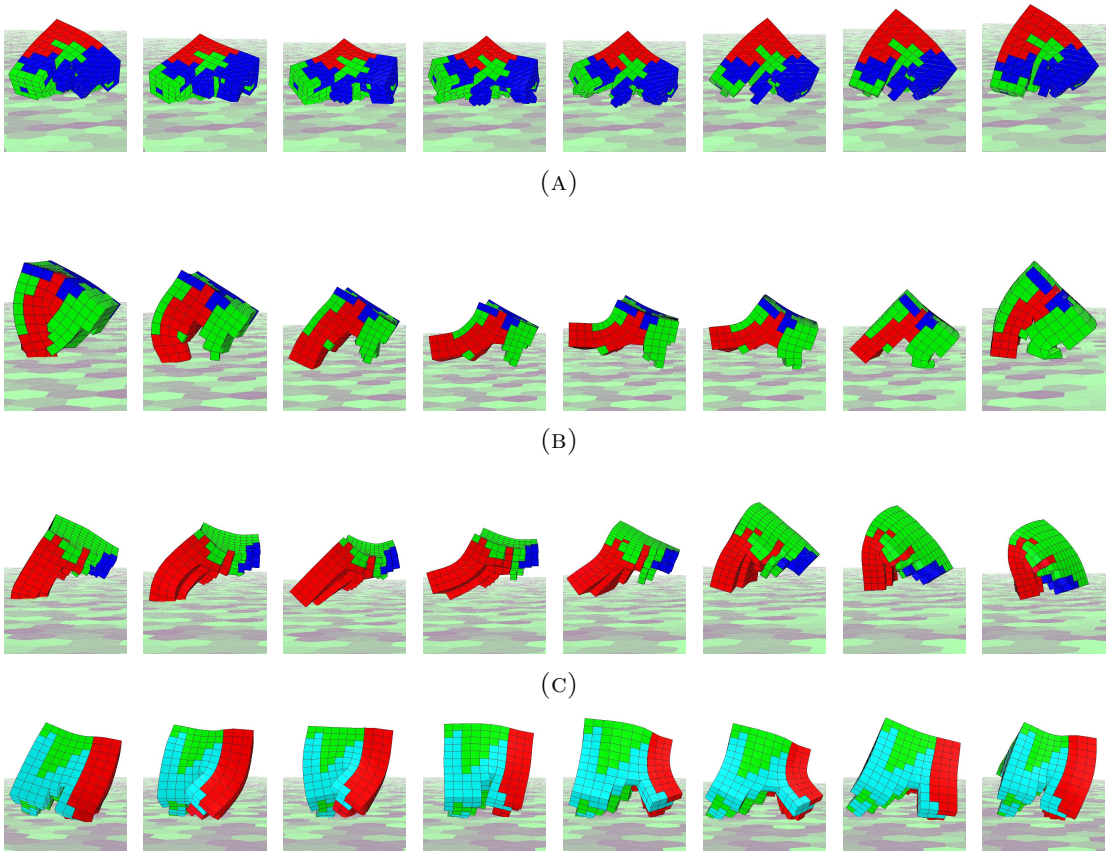


FIGURE 5.1: Champion (best overall) morphologies evolved in independent runs within fitness-based search. Each row presents the locomotion strategy of the individuals created.

5.1 Evolved Morphologies



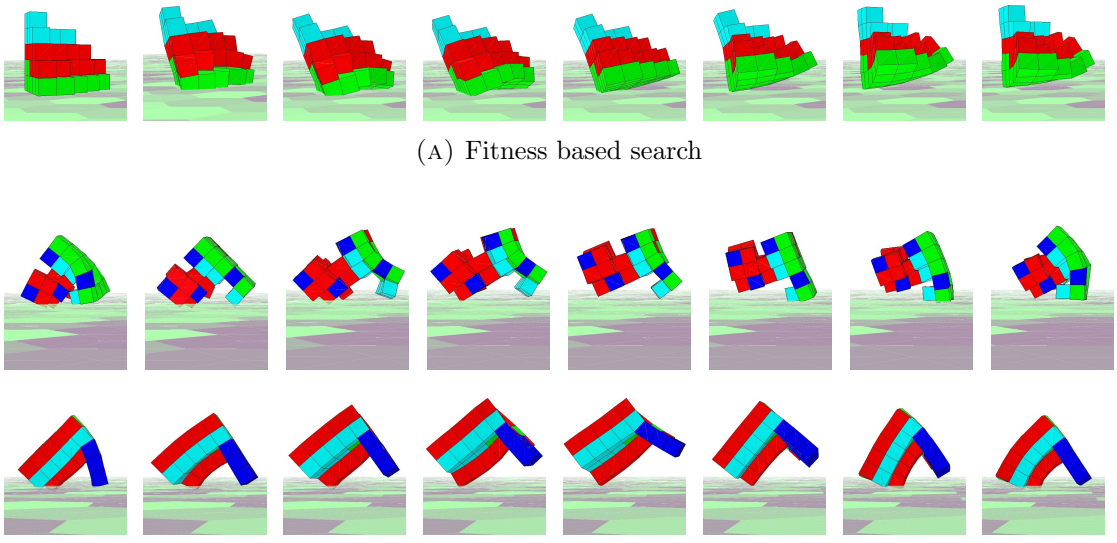
(A)

(B)

(C)

(D)

FIGURE 5.2: Champion morphologies evolved in independent runs within novelty search. Each row presents the locomotion strategy of the individuals created.



(A) Fitness based search

(B) Novelty search

FIGURE 5.3: Champion morphologies evolved in independent runs within fitness-based and novelty search. Each row presents the locomotion strategy of the individuals created.

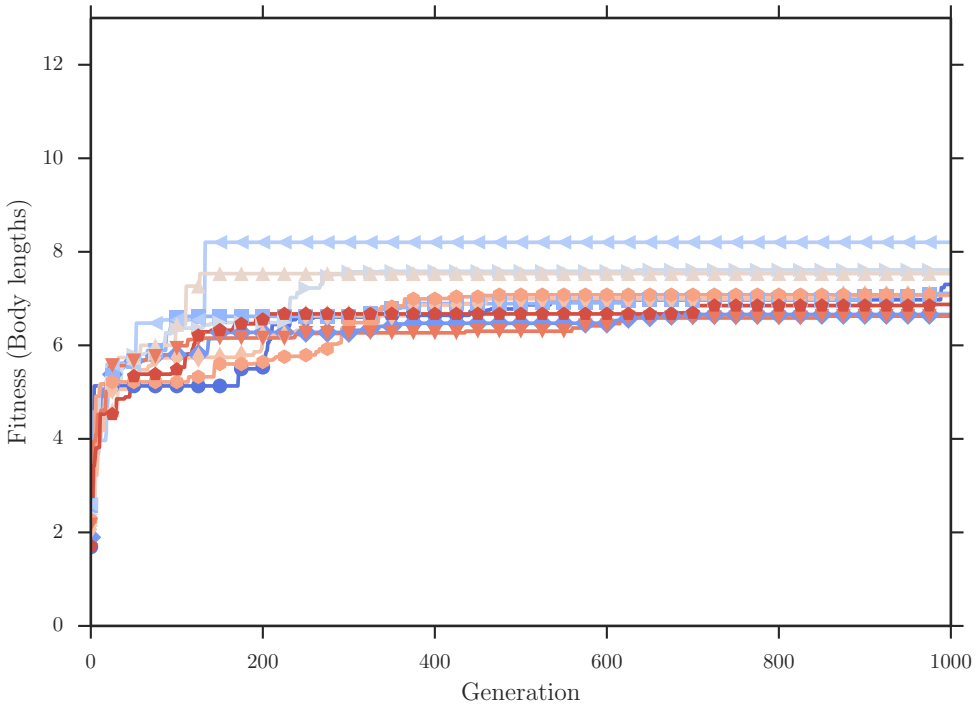


FIGURE 5.4: Best so far fitness, 10 individual runs for fitness based search (settings A.3.3).

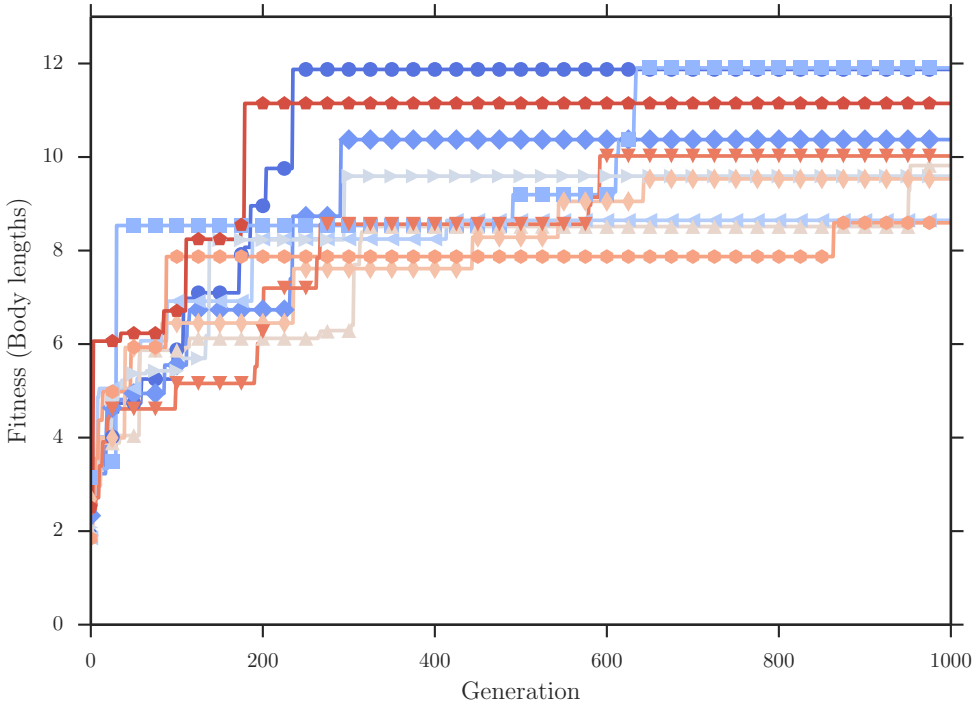


FIGURE 5.5: Best so far fitness, 10 individual runs for novelty search (settings A.3.3).

5.2 Into The Performance of Novelty Search

Before compare novelty search to fitness based search, it is of interest to show how individually perform under the same simulation settings.

Figure 5.4 shows 10 independent runs for fitness based search. Following the objective function's gradient fitness based evolution does small step towards better and more optimized individuals from generation to generation. What is more, fitness based evolution often sticks into specific morphologies which then tries to optimize leading the evolution to stop at that local maximum.

Figure 5.5 shows 10 independent runs for novelty search. In comparison with the same figure for fitness based search (fig. 5.4 we can see a clear difference. Evolving for novelty means that within the evolution only a novel behavior is rewarded instead of a good behavior or a behavior that leads to the optimization of the objective function. Big steps in the fitness value on all independent runs can be observed which can lead us to a conclusion that fit individuals in respect to the objective function for which novelty search has no information within the evolution process, are results of new novel behaviors.

One more thing worth noticing, is that observing only big steps in the fitness, we can derive that there is no optimization of morphologies within novelty search. Initially, novel individuals are highly rewarded, these individuals can be very good in respect to the fitness or not, the algorithm does not consider the "goodness" of these individuals, and does not have any information regarding this either. On the next generation, mutations, crossovers, and copies of these novel individuals are not going to be highly variant in respect to their chromosome from their ancestors, resulting to similar behaviors, which are not going to be remarkably rewarded in respect to their novelty value. Thus, highly novel individuals are producing less novel children, meaning that these children, even though their fitness is high and can be optimized further, will not have the chance to reproduce in the next generations and be improved eventually, as in fitness based evolution.

To extensively compare the performance achieved by novelty search method the same experiment held under two different simulation settings (for sizes 5^3 ,10^3), set side by side with fitness search, random search, and finally a simple genetic algorithm. Notice, that the first three methods are referring to a generative encoding (CPPNs) evolved by Hypercube NEAT evolutionary algorithm and using selection in respect to fitness, novelty and finally random selection, while the latter uses a direct encoding scheme driven by fitness.

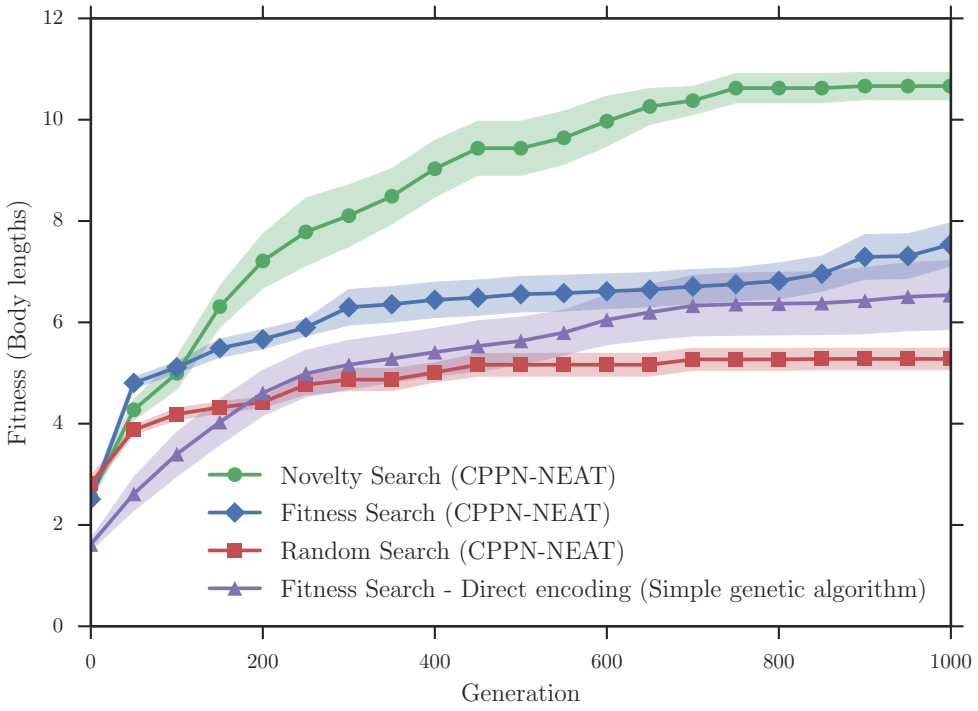


FIGURE 5.6: Comparison of simple genetic algorithm (direct encoding) against *random* - *fitness* - *novelty* search with generative encoding. Best so far fitness averaged over 10 runs (settings A.3.1).

Novelty search to perform the novelty metric computation, makes use of the two dimensional trajectories, which are all normalized so that their centre of mass of the trajectories coordinates meet a specific angle, as well as their starting coordinate is always located in the beginning of both axes. Fitness-based search objective function is the displacement of the soft-robot's center of mass from its initial position in body-lengths. Random evolution with Hypercube NEAT achieved using random selected individuals to breed. For direct encoding, the method used is has been explained in Chapter 4.

Figure 5.6, presents the results for the small sized structures (5^3). Notice, the difference between novelty search and fitness-based method, novelty evolves structures that are superior than any other method does in these settings. At this point, it should be mentioned that in such a small structures locomotion patterns cannot be evolved due to the stability issues of the simulator, and due to the fact that lightweight structures can be bouncy, leading to ball shaped structures capable of achieving large displacement from their initial positions. That being said, we still have to deal with an optimization problem, where local optima and global ones can be found as the number of the possible solutions in this setting, using 4 materials, is $\sim 2,3 \times 10^{87}$. Using the trajectories of the soft-robots, novelty search visits optimal solutions that none of the other methods does because of local optima (fitness-based search), due to encoding limitations (direct

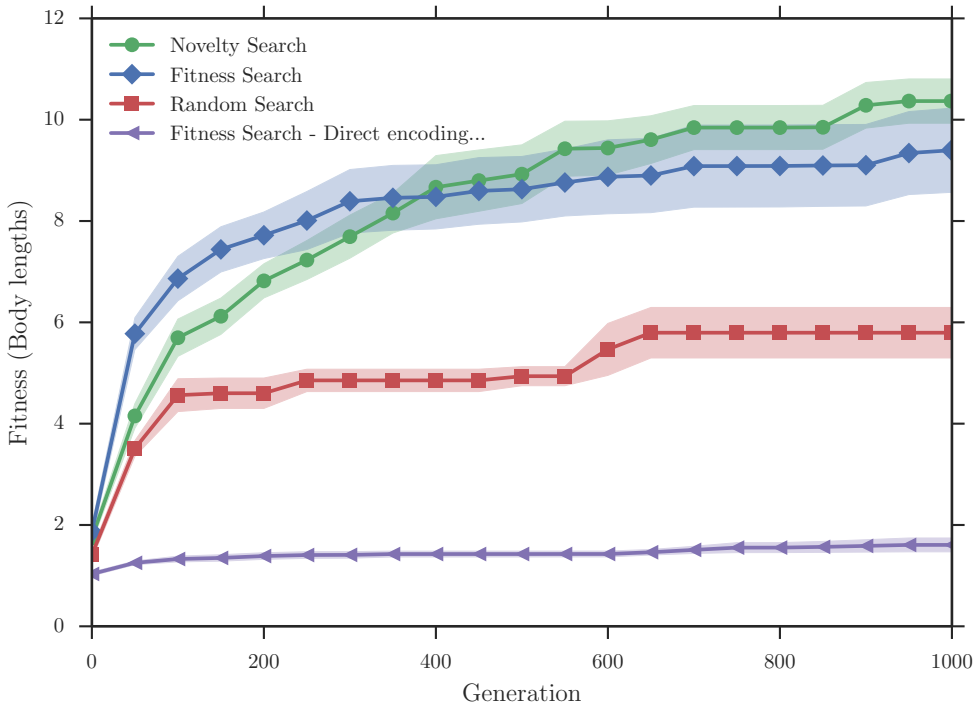


FIGURE 5.7: Comparison of simple genetic algorithm (direct encoding) against *fitness - novelty* search with generative encoding. Best so far fitness averaged over 10 runs (settings A.3.2).

encoding), or random search which selects random individuals to reproduce without caring about their performance, and with no backtracking (there is no guarantee that random search will visit new behaviors). The simple genetic algorithm approach which uses a direct encoding to represent the structure of the soft-robots performs better than using random selection within an indirect encoding evolution pointing out that symmetry does not provide any merits to the evolution for these sizes soft-body robots.

Moving to a larger lattice space size we expect indirect encoding to prove its advantages over the direct encoding scheme [6, 40]. Furthermore, novelty search now has a more difficult task as the space of possible behaviors (2d-trajectories) becomes larger as more complicated morphologies can now be produced (morphology space for 10^3 lattice space: 9.3×10^{698}). To try to solve all these research question the same experiment held under a lattice of size 10^3.

Figure 5.7, presents the results of the same four different methods in this setting. Results reassure that novelty search achieves higher fitness on average against fitness-based search. Nevertheless, there is no tremendous difference as in the previous experiment, discovering that at their individual runs they both achieve to evolve the soft-robot structure with the highest fitness found in all experiments ($\sim$ 14 Body lengths). Novelty search seems more constant in evolving individuals with high fitness in all runs, on the

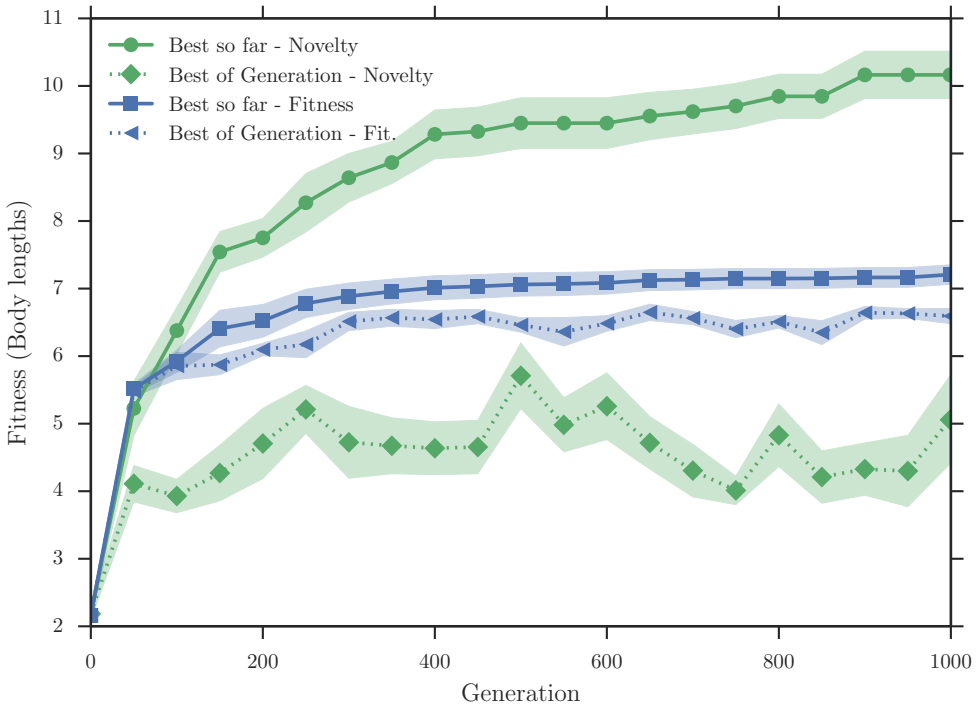


FIGURE 5.8: Fitness of the generation's champion (best individual) for *fitness - novelty* search averaged over 10 runs (settings A.3.3).

other hand most of individual runs of fitness search trapped in a low fitness local optimum structure, trying to optimize the specific individual genotype without trying to explore more the fitness landscape like novelty did successfully. Once again, random evolution with Hypercube NEAT is producing decent structures for soft-robots but cannot climb the hill of fitness, going in every direction, at the same time making more and more complex network topologies for CPPNs. Earlier in this thesis, in chapter 2, generative encoding advantages over direct explained in detail, here their superiority can evidently be observed. Direct encoding performance when a larger lattice for the simulation used, was radically decreased, mostly because structure and morphology regularity is a necessity for the soft-robots in order to perform decently in these sizes, something that direct encoding cannot capture failing to produce anything useful.

Another aspect of the evolution should be inspected is how the population of each generation is affected in respect to the best individual per generation, especially thinking about the these generation champions is the ones that result in the increased of novelty search when compared with fitness based search. In figure 5.8, the champions' fitness (Best fitness found within each generation) of each generation is plotted averaged over 10 runs. Recall, that novelty search does not have any information about fitness of individuals. In fitness based search there is a clear trend that champions of each generation are getting better through the evolution resulting to an approximately monotonically

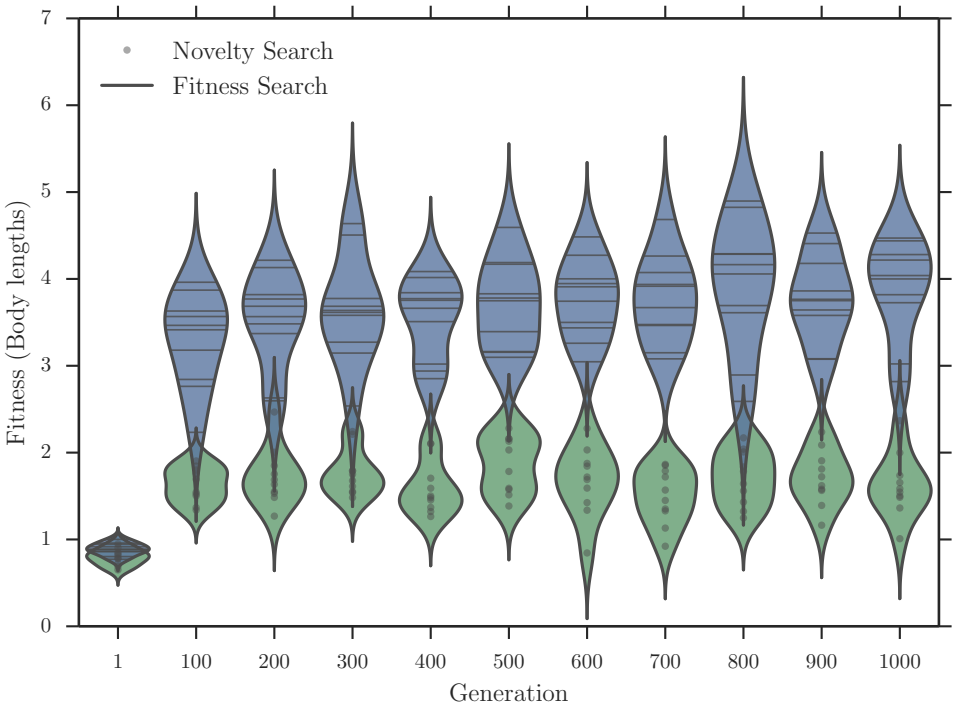


FIGURE 5.9: Distributions of average population fitness per generation over 10 runs for *fitness*(Blue) - *novelty* (Green) search with generative encoding (settings A.3.3).

increasing function. On the other hand, generations' champions in novelty search apart from the early improvement which is mainly caused by the generative encoding, follow a random pattern. What it is interesting here to see is that even though that the solutions novelty search gives, in this settings (lattice size: 7^3), are clearly better than the ones evolved by fitness based search, on average the champions during novelty search evolution are worse. Hence, individuals that resulting in the increased performance of novelty search clearly lie on the tail of the fitness distribution on each generation.

In the same fashion, the average population fitness seems also affected by the different optimization methods. Figure 5.9 illustrates the distribution of population's average fitness over 10 independent runs for *novelty-fitness* based search every 100 generations. The resulted distributions which are shown in violin-like shapes clearly show that the average generation's fitness remains stable through the whole evolution (1000 generations) for both methods. What is more, the generation's average fitness is significantly lower for *novelty* search, meaning that when the evolution is being carried towards novel behaviors there is no such guarantee that assumes novel new founds in the behavioral space will also be *fit*.

What we see in the last two figures, evidently shows that even though novelty search achieves in finding more "fit" solutions than fitness based search in the specific problem

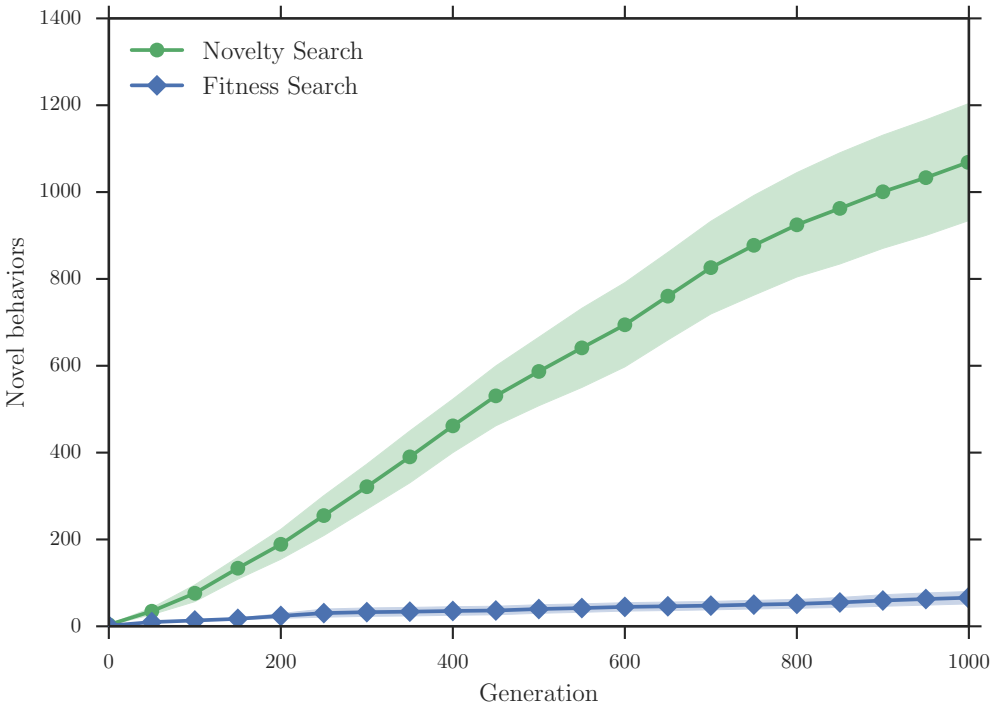


FIGURE 5.10: Number of novel behaviors found up to generation number, averaged over 10 runs. The novelty measure is computed as the average distance from the 10-nearest behaviors for *fitness* - *novelty* search with generative encoding (settings A.3.3).

domain, the average fitness of both generation champions and population remain lower than in fitness based search.

Until this point, the performance of both fitness and novelty search methods have been compared in the same objective metric such as the displacement of the produced soft body robots. The former method method tries to optimize genomes in respect to the specific objective function, while the latter moves its interest into creating diversity of the population in the behavioral space. As shown before, the novelty search achieves to create novel individuals which are not only novel in respect to how different behaviors they have from the rest of the population they exist into, but also they achieve higher average fitness than those they are optimized towards that objective. Inverting the objective function now such as our goal is to generate a wide variety of behaviors, in this case, two dimensional trajectories, we expect that a much larger set of novel behaviors will be created by novelty search. Figure 5.10, presents the number of unique behaviors the two evolutionary methods found, averaged over 10 runs. The resulted graph shows that comparing these two methods is pointless as *novelty* search can force the evolution towards spaces in the behavioral space that have not visited, finding more novel individuals, which does not happen in the fitness search. Surprisingly, *novelty*

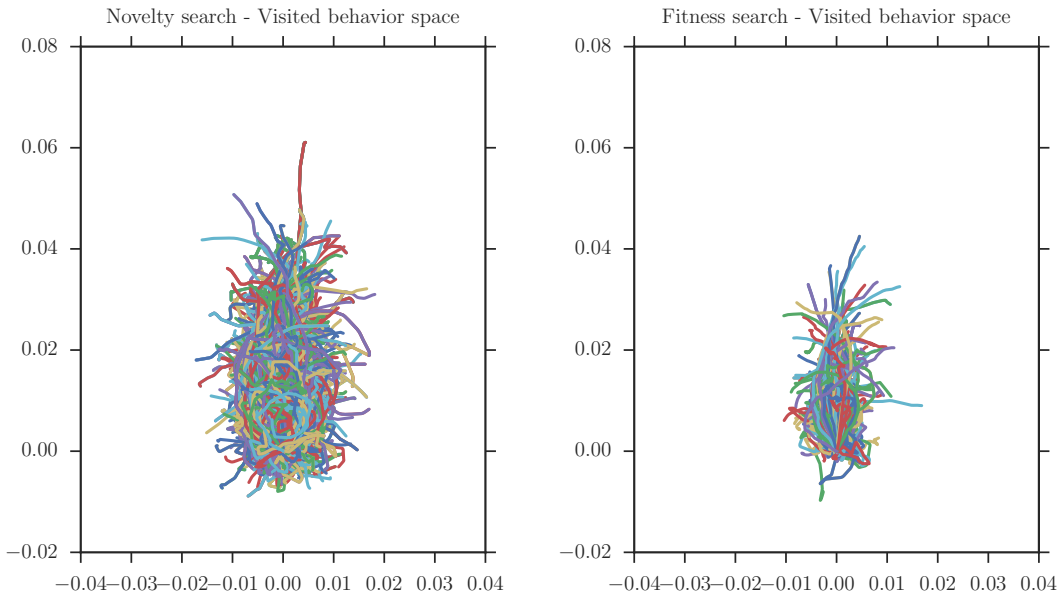


FIGURE 5.11: Novelty search creates a vast amount of behaviors achieving in this way to find fit individuals, and avoid local optima of the solution space. (settings A.3.3)

achieves better performance than *fitness* search in both objectives set so far, creating fit, and at the same time diverse solutions.

To visualize the difference in the behavior space of the two methods, figure 5.11, illustrates all the stored found novel behaviors (two dimensional trajectories) found in one evolution run of novelty and fitness search using the same novel measure to determine the novelty of a behavior.

5.2.1 How Behavior Selection Affects *Novelty*-Search

A good behavior metric should include information about the objective function. In case of locomotion gait of soft robots a trajectory can be highly informative as far as the displacement of the robot's body, as well as the gait, is concerned. Two robot bodies which travelled the same distance into an equal time horizon, should have the same fitness if displacement is only measured, nevertheless, the locomotion strategy, is something that can only affect the actual behavior metric and not the objective function. Forcing the evolution to seek for the novel in the behavior space results in producing $> 10\times$ more novel behaviors than *fitness* search (depending on the threshold and the behavior metric), which indirectly implies that high fitness individuals will be found as the behavior space is heavily searched. How the behavior metric affects the performance of the evolution is discussed in detail in the next section.

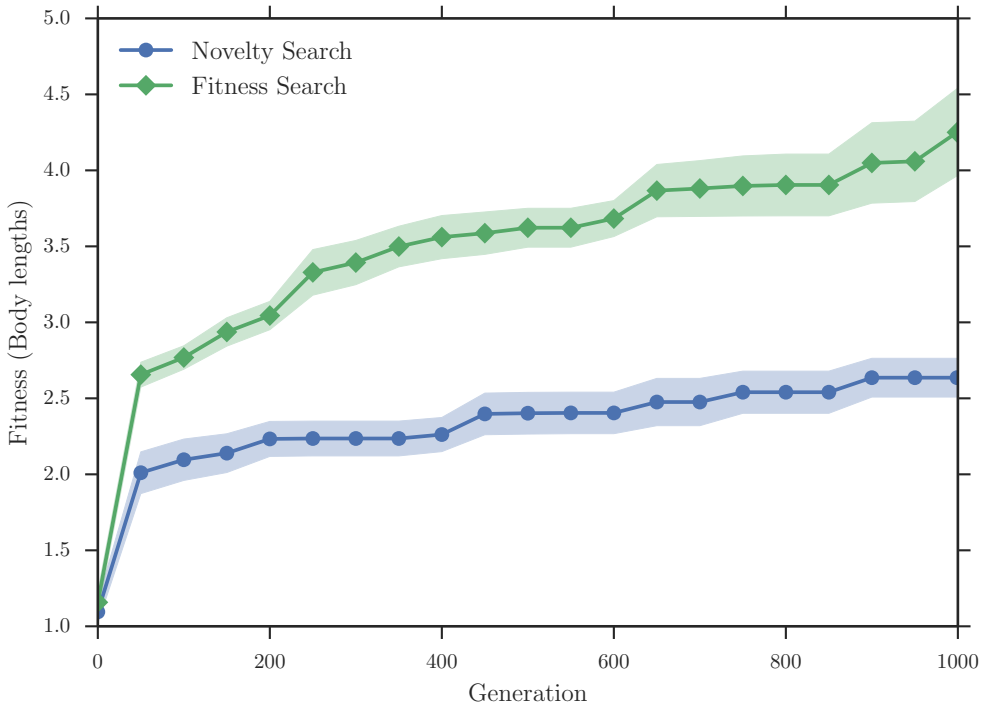


FIGURE 5.12: Best so far fitness averaged over 10 runs, penalizing actuated materials[1] for *fitness - novelty* search with generative encoding (settings A.3.1).

The importance of selecting a good behavior metric is important in order for novelty search to explore the behavior space to a great extent. For example, searching for fast robots while you exploring the behavior space of their trajectories is a wise decision considering that all information needed to determine the fitness (speed) is incorporated inside the behavior (trajectories) assuming static sampling rate of the trajectories. In this experiment to investigate what is the result of the novelty-search evolution when no information about fitness is provided by the behavior, a objective function was selected that the currently used behavior metric doe not include information about. The two dimensional projection of the trajectories in x, y-axis are again selected, while instead of evaluating the fitness in displacement, this displacement is penalized by the number of actuated voxels are inside the structure of the soft robot. Figure 5.12, illustrates the best so far fitness for both novelty and fitness search averaged on 10 independent runs. Comparing the results with figure 5.6, one can notice how novelty search performs poorly in this setting. Considering that the same method outperforms traditional fitness-search evolution when the whole information of the fitness function is contained in the behavior. Trying to find novel trajectories in the first case proved successful in respect to the final

[1] Actuated materials penalize fitness:

$$f = (1 - (n_{actuated}/n_{total})^{1.5}) \times disp$$

, where $n_{actuated}$, is the number of actuated voxels, n_{total} total number of voxels and $disp$ the displacement of the softbot's center of mass.

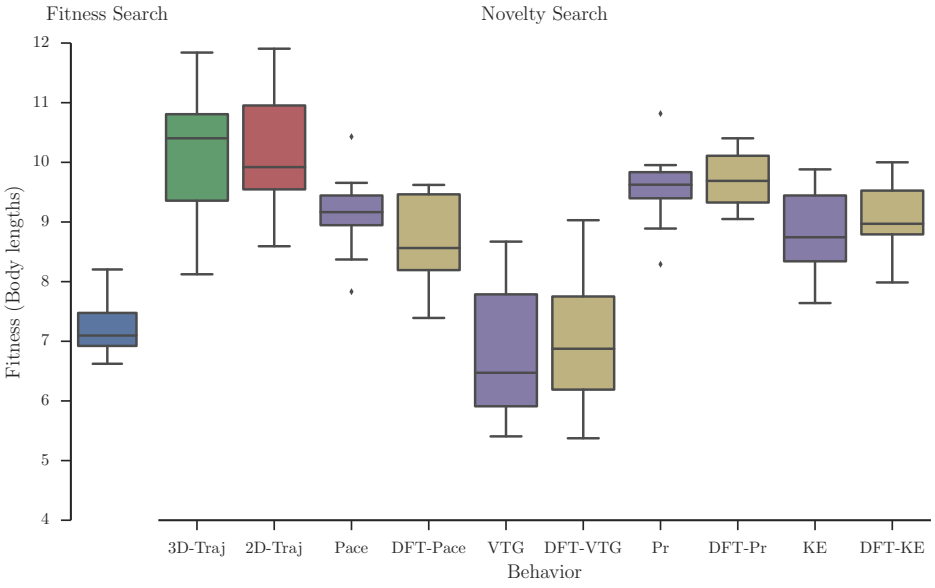


FIGURE 5.13: Comparison of the evolution's best fitness result from 10-runs under different behavioral metrics for *novelty* search (right). *Fitness* search is also evaluated under the same settings (left - blue box). (settings A.3.3).

displacement of the individuals produced. On the other hand trying to maximize the distance and in the same time use as few actuated voxels as possible, proved crucial for the final outcome of the search for novelty. If the number of actuated voxels had been included in some way into the behavior metric, novelty-search would have been more exploratory towards this direction as well.

Choosing the appropriate metric to describe a phenotype into the behavior space is crucial in the performance of novelty search. Figure 5.13 illustrates the experimental results under different behavior metrics, alongside the performance of pure *fitness* based optimization. A set of 10 different behavior types was used including the three dimensional trajectories of the soft robots (3D-Traj), the two dimensional projection on x, y-axes of the previous behavior (2D-Traj), the pace sampled every 0.001 sec. (Pace), the discrete Fourier transformation of the same signal which was sampled every 0.00001 sec. (DFT-Pace), the voxels touching the ground on each time-step (VTG, DFT-VTG), the maximum pressure per time-step (Pr, DFT-Pr), and the kinetic energy of the whole structure (KE, DFT-KE). What is shown here, is the fitness in body lengths of the champion individual during the whole evolution from 10-independent runs of the experiment. Both trajectory behavior types achieve the best performance as far as fitness is concerned, with a small difference in favor of the three dimensional one. Pressure is coming third achieving high performance close to the previous two trajectory behavior types, pace and kinetic energy of the structure are next in the performance ladder, and last one is the behavior signal that count how many voxels touch the ground on each sampling time-step. The results of using 10 different behavior types can be clustered into

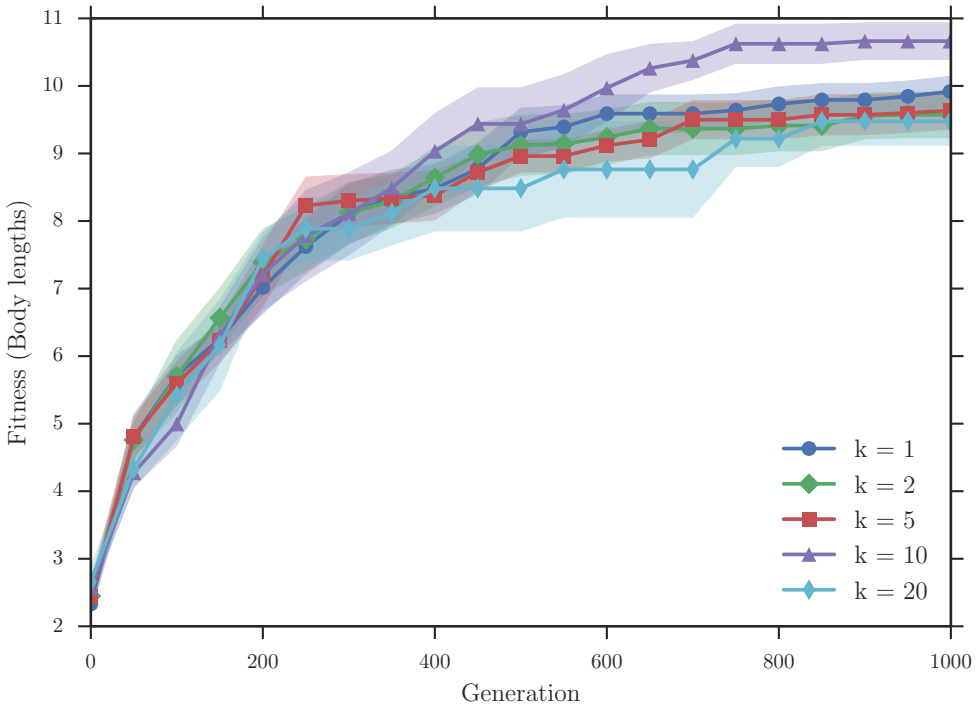
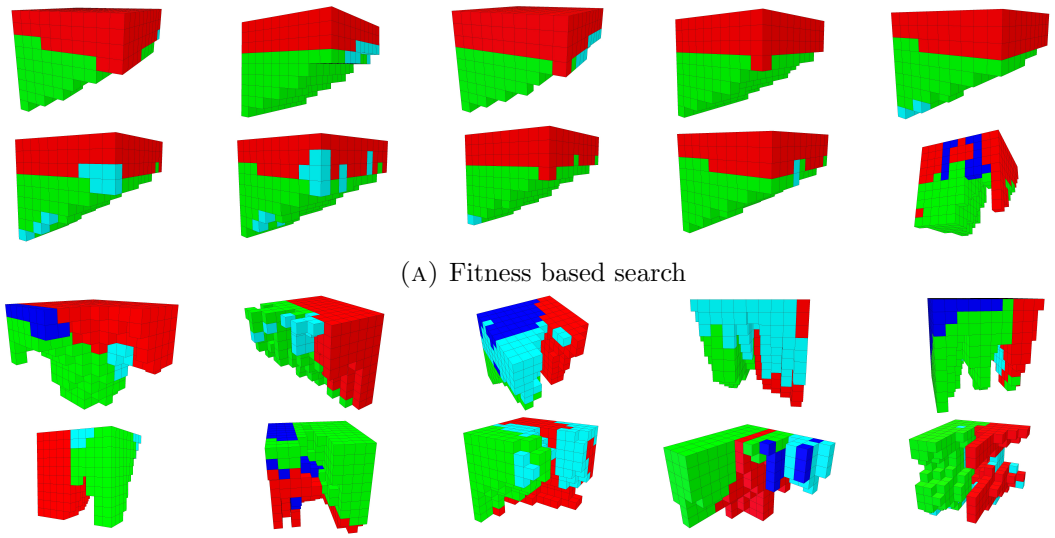


FIGURE 5.14: Best so far fitness averaged over 10 runs, for different k to sparsity computation of the behavior (settings A.3.1).

three performance categories. The first one which includes the two types of trajectories and achieves the best performance of all, the second one which includes raw values and the discrete Fourier transformation of pace, pressure and kinetic energy, the last and worst one with the number of voxels touching the ground.

The performance of novelty search when trajectory of the soft bodies is used as a behavior metric is superior over all other behavior metrics. Trajectories are a very good selection for this kind of problem, since they can indirectly not only encode the objective function which is the displacement, but also the locomotion strategy and that is the reason why they explore better the landscape of behaviors resulting in such high difference in fitness against the fitness search.

The rest of the behavior metrics apart from VTG and VTG-DFT, are close, as far as the final performance of the evolution is concerned. On reason that they fail to meet the trajectories performance is the fact that even though they keep track of features that can actually measure the performance of the robot, speed, they cannot encode the direction of the soft-body during the simulation. Which is crucial if we think that soft-robots having a circle trajectory, even though they can produce fast locomotion that displacement from their initial positions will remain low.



(A) Fitness based search

(B) Novelty search

FIGURE 5.15: Fitness based search trying to optimize a specific structure while the search for novelty results in a variety of shapes.

Counting the number of voxels in a structure that touch the ground in every timestep of the simulation, does not have any implication about how fast the robot is moving. A fast moving robot that is hopping can have the same behavior signature with a hopping robot that stays in the same position after each jump, yet, using the trajectories these two soft-robots will have a huge difference in their behaviors.

Within the same figure and on the left side of it (blue box), *fitness*-search is also evaluated under the same experimental settings. The performance of this objective optimization method is only comparable with the worst *novelty*-search scenario when the VTG behavior is selected for novelty to be measured.

5.2.2 Sparsity in *Novelty*-Search

Sparsity (eq. 2.1) is a measure that defines if a newly found behavior is novel enough to enter the set of novel behaviors. Figure 5.14 presents the resulted best so far fitness given different values for $k \in \{1, 2, 5, 10, 20\}$. In principle k can define how tolerate the algorithm can be with new behaviors. It is not certain that a specific value for k should give the highest performance in fitness and it depends almost completely by the application. The only implication in choosing value for k is that choosing large values should yield in a more detailed exploration in the behavior space, in the contrary using small values final set of behaviors will be denser in the behavior space. In the specific figure and experiment $k = 10$ was the setting that led to the best performance.

5.2.3 Diversity of Individuals in *Novelty*-Search

Figure 5.15, shows the champions every hundred generations of a experimental run for novelty search and fitness based search. While the fitness based search is stuck trying to optimize a specific morphology of a soft robots, novelty search is taking a walk in behavior space unveiling new morphologies for the soft-body structures. The same motif appears in every independent run of fitness and novelty search, while novelty search achieves the a variety of morphologies fitness search is sticking to certain shapes, different in every run. It is obvious, that both search methods have their advantages and disadvantages. First, fitness-based search optimizes (optimized distribution of material within the structure) certain shapes during the evolution, at the same time novelty does not optimize them. Novelty search because it is a diversity based method, new shapes are evolved but not optimized. Next section discusses ways of combining both search methods merits.

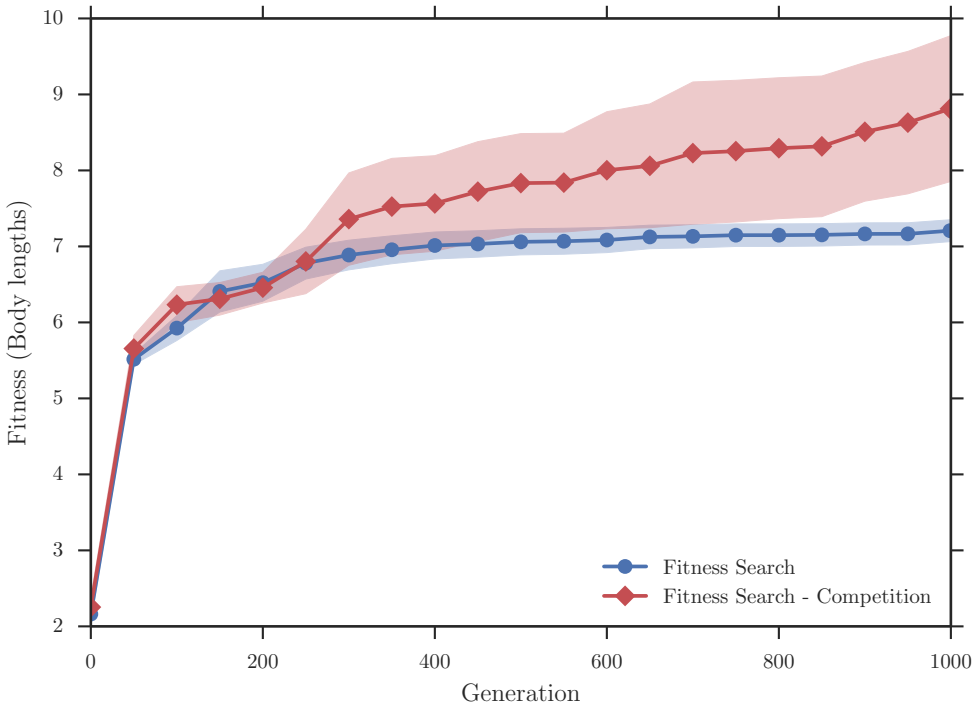


FIGURE 5.16: Best so far fitness averaged over 10 runs, with no competition, local competition in the complete population of each species for *fitness* search (settings A.3.3).

5.3 How Selection Affects the Performance of Both Search Methods

Discussed extensively in the previous section, selection is a process that picks individuals in order to breed, be mutated or copied into the next generation. It is the part of the evolutionary algorithm that is responsible for producing the new generation, based on the individuals which exist into the current one.

Fitness Search

Figure 5.16, presents the results for two different selection methods, random selection from the top 20% (Blue) and competition within individuals from the complete current population (Red). As it was expected, competition, as well as, the fact that the whole population has the opportunity to breed, contribute to the diversity of the population. This can be easily seen in this figure, random selection within the top 20% of the population does not allow solution to reproduce meaning that it does not explore weaker individuals, which can later after enough mutations become better than the potential of the rest of the population. The deviation of the first method gives a perfect clue about how narrow is the fitness landscape at the converged area of search when only the best of each generation are allowed to breed.

Novelty Search

Since the algorithmic framework is the same for both searches, competition can be applied in novelty search as well. Figure 5.17, presents the results when competition is held among individual of the whole generation's population among species in respect to different metrics. Competition is held among individual regarding their novelty among the whole population of the evolution and the novelty value they obtain if they are only compared with their species population. In both cases the overall performance of the evolution averaged on 10 runs is worse than the default setting in novelty search where individuals to breed are selected randomly from the top-20% of the population of each species. Both selection approaches are performing poorly set side by side with the default selection method. Selecting individuals with high novelty withing the species is crucial for the performance, since these individuals can have low novel value when compared with the global population, leading to steps backwards in the evolution towards highly novel individuals. On the other hand, when individuals are competing using their global novelty measure leads to a slightly better performance, still far from the default setting, meaning that highly novel individuals can actually produce more novel individuals when

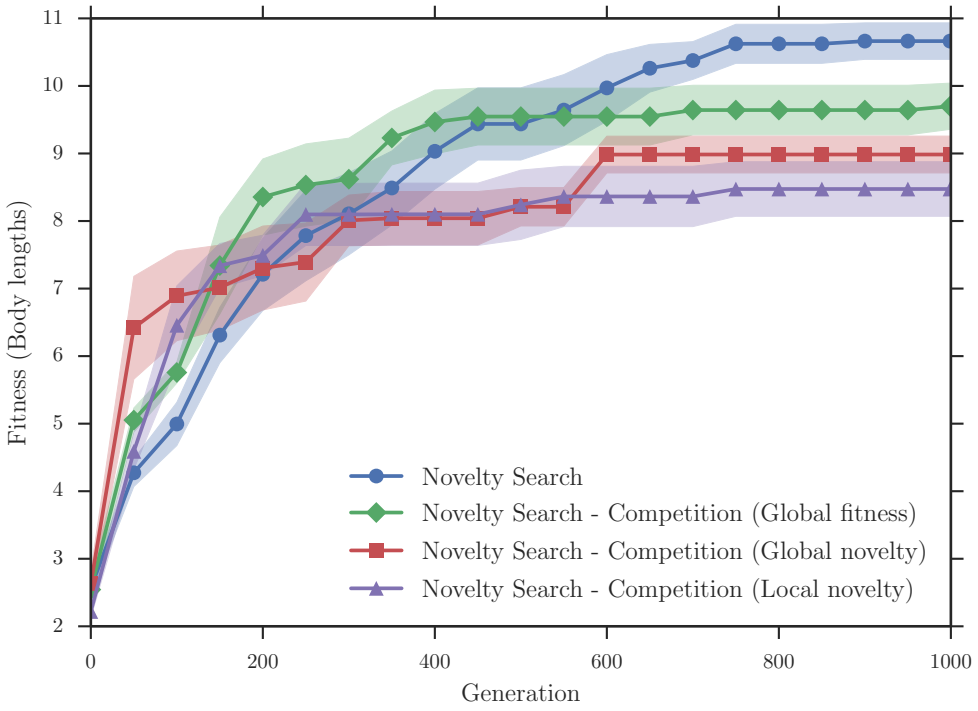


FIGURE 5.17: Best so far fitness averaged over 10 runs, for local competition held among the population of each species for *novelty* search with generative encoding (settings A.3.1).

they are allowed to breed. In other words, competition disturbs the properties of novelty search, while in fitness based search merits of selecting not only the fittest individuals were shown.

5.4 Incorporate *fitness* Information into *Novelty*-Search

The reason that novelty search is considered such an revolutionary search method is because it finds solutions for deceptive problems, where the fitness landscape is not a straightforward function. What makes it so unique, it is the fact that instead of looking for better solutions in respect to an objective function is looking for different solutions. In each generation of the novelty search there are some solutions that are very good regarding their objective function value, eventually these novel individuals will stop being selected for reproduction since their novelty metric value will be declined as more individuals with similar behaviors will be produced by mutations of the same novel individuals, and they won't be optimized as they could have been. Mutations and other genetic operations can optimize these fit individuals more, but this is something that happens in fitness-based search. These individuals (with high fitness value) can be seen as *stepping stones* [12] towards more optimized versions of them. Being blind

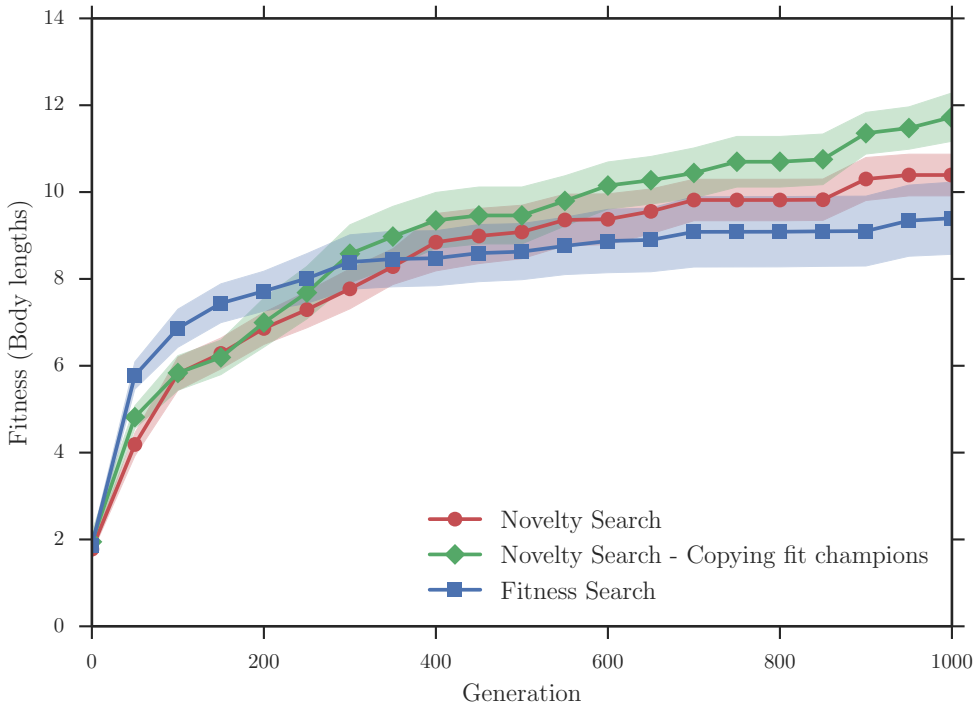


FIGURE 5.18: Best so far fitness averaged over 10 runs, for *novelty* search with and without copying *fit* champions and *fitness* search (settings A.3.2).

to the objective function, novelty search will eventually stop producing new individuals out of them, which will lead to promising individuals being thrown out of the evolution process.

Competition is a simple way of combining the two searches together, after each generation is produced competition is held over all the population within each specie, selecting individual, for reproduction, that have high value in the objective function. Figure 5.17, illustrates the results of using the global fitness as a measure for selection among two generations. The results (Green line), reveal that competition for fitness in a novelty search setting disturbs the balance of the evolution towards novelty, not allowing novelty search to expand the search in a greater extend, since it is not the case that selected fit individuals will lead in novel behaviors.

Fitness Elitism in Novelty Search

It has been shown, how selecting individuals in respect to their fitness distracts the evolution in novelty search. Hence, a new method is proposed for incorporating fitness information into novelty search without perturbing with its pipeline. Elitism is the process of copying the best individual of each species into the next generation with a probability of mutating it first. In this way best individuals are preserved and can

be optimized later, which considered to be a successful way of protecting the best of each species generation so they can contribute with their beneficial genes later in the evolution. Novelty search can include elitism in its selection process, and it does that by copying the most novel organisms of the current population of each species to the next. Since, there is no point of changing this function, elitism can be used also to copy fit individuals within novelty search method.

The way these two elitism functions can be used depends on the population size, and the problem. Probabilistic methods can also be used combining both elitism functions. In the specific setting, both elitism function copy new individuals to the new generations. In this way the evolution towards novelty does not get disturbed, at the same time, highly fit individuals have the chance to be optimized further as long as they are the fittest within the species population.

Figure 5.18, illustrates the gain in performance when fitness elitism is used in novelty search method compared with pure novelty and fitness based search methods.

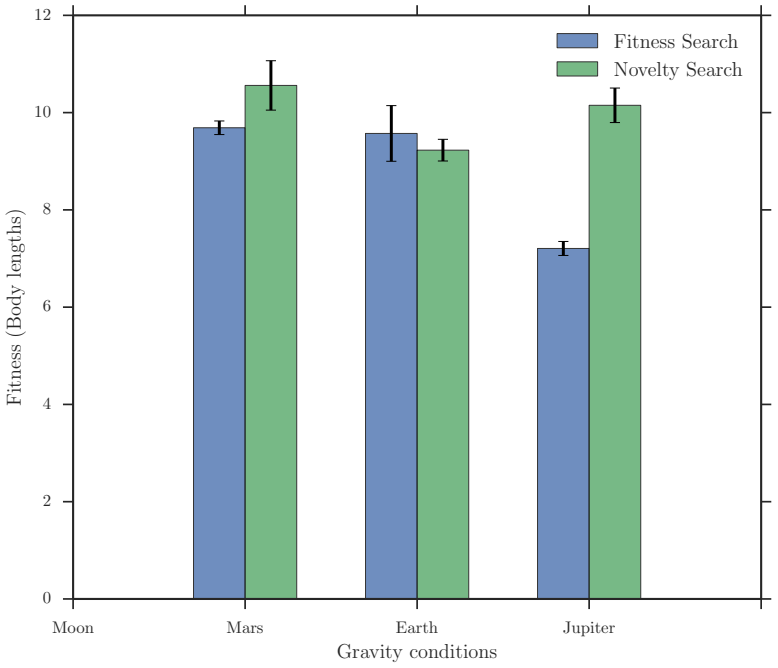


FIGURE 5.19: Novelty search performs equally good or better than fitness based search in all gravity conditions tested. (settings A.3.3)

Experiment not finished yet

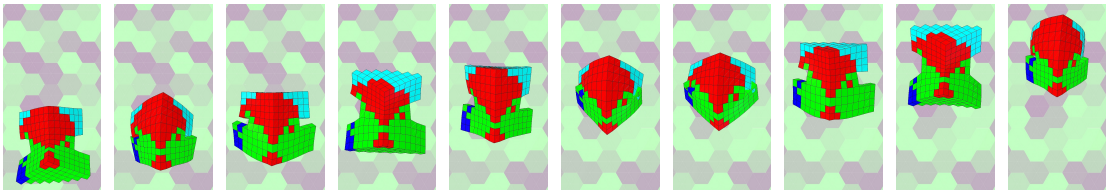
5.5 Evolving Soft-Robots for Outer Space

In this section, it is of interest to show how different environmental conditions can affect both the search and the type of locomotion produced by the evolved soft robots. Since, similar conditions of other planets into our solar system are difficult to be reproduced by the simulation environment is used, we only interested to replicate the previous experimental settings with variant gravity acceleration conditions. Fitness based and novelty search are used again within the CPPN-NEAT evolutionary algorithm to evolve the morphology and the locomotion strategies of these voxel structures. For the novelty search two dimensional trajectories of the soft bodies are chosen as the behavior metric to evaluate the novelty of each.

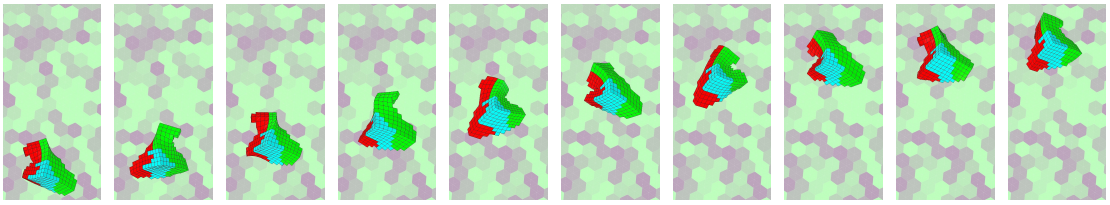
Figure 5.19, illustrates the performance of these two search methods, in four different settings, each method for more than five independent runs, the best fitness achieved by an individual averaged for all runs are shown together with the deviation errors. Default setting used in all previous experiment used for Jupiter's and Earth's gravity accelerations, while the simulation time used for both was 0.4 seconds. For Moon's and Mars' evolution runs, a higher temperature period was used (0.050 seconds), in order effective locomotion to take place, as higher frequencies tend not to be able to

produce any locomotion in lower gravity conditions. Furthermore, for the latter two gravity levels, the simulation time was larger up to 1 second for each evaluation, as the velocities generated by the soft-robots were significantly lower in such low gravity levels.

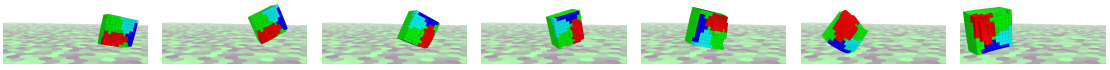
discuss results



(A) 2-legged galloping (Fitness-Search)

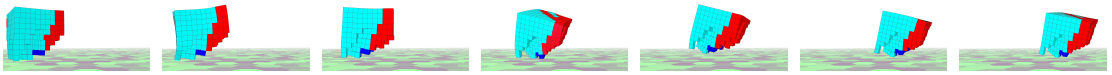


(B) 4-legged animal like locomotion (Fitness-Search)

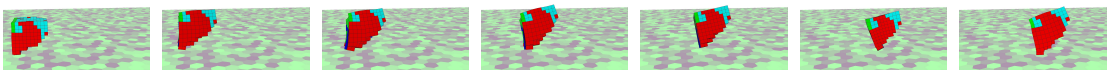


(C) Tumbleweed-like locomotion (Novelty-Search)

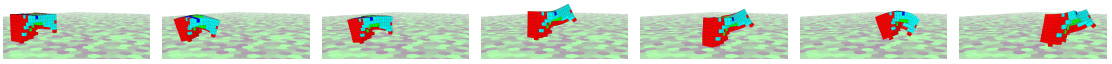
FIGURE 5.20: **Earth**: Morphologies evolved in gravity conditions on Earth, show that life-like locomotion strategies can be generated by soft-body creatures in a simulated environment.



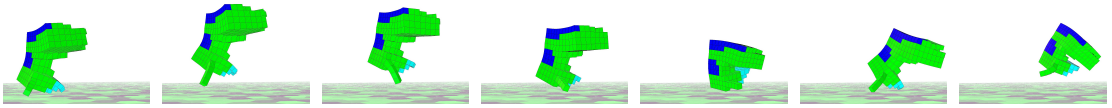
(A) 3-legged hopper (Fitness-Search)



(B) 1-legged hopper (Fitness-Search)

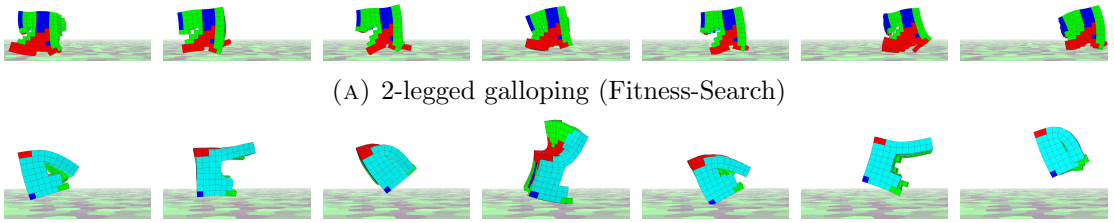


(C) L-shaped hopper (Novelty-Search)



(D) C-shaped hopper (Novelty-Search)

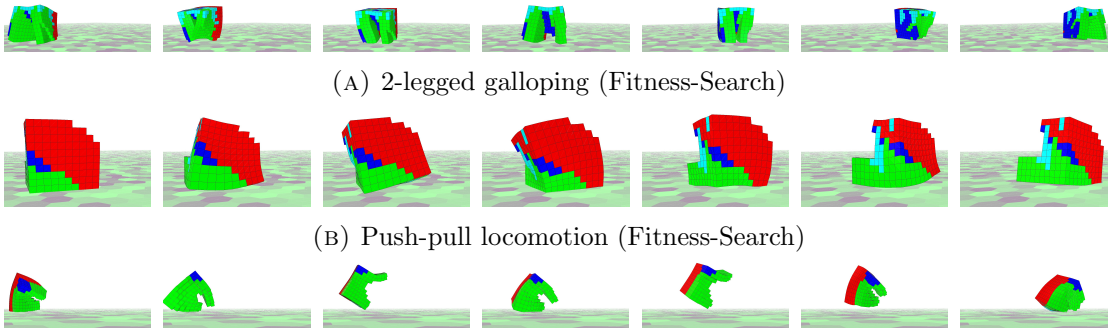
FIGURE 5.21: **Moon**: Locomotion strategies evolved in low-gravity conditions (Moon) consist mostly of hopper soft-robots.



(A) 2-legged galloping (Fitness-Search)

(B) 2-legged C-shaped hopper (Novelty-Search)

FIGURE 5.22: **Mars**: Gravity acceleration on Mars allows both galloping and hopping locomotion strategies.



(A) 2-legged galloping (Fitness-Search)

(B) Push-pull locomotion (Fitness-Search)

(C) C-shaped hopper (Novelty-Search)

FIGURE 5.23: **Jupiter**: Heavier structures on Jupiter's gravity level can locomote efficiently using several strategies.

Chapter 6

Future Work

This chapter discusses possible future research that can be done in the topics approached and the contributions of this thesis. Designing soft-body structures in a simulated environment is a heavy task for human designers [40], evolutionary algorithms [10] and generative encoding [6] combined, succeeded in the evolution of these designs, as well as, and the coordination (distribution of materials) of the structures evolved. What follows in this chapter is points worth further investigation in the science of evolutionary soft-robotics, still in a simulated environment.

Evolution of materials

The scope of this thesis includes the evolution of soft robots morphologies given a set of predefined materials with specific properties. The reason behind this, is the fact that is was only of interest to investigate ways of designing soft-robots having specific materials as building blocks. Another aspect in the evolution of the morphology of soft robot bodies is the evolution of the materials alongside the structure. The possibility of a dynamic palette of materials will enable more complex gaits for the soft bodies evolved. Using the same generative encoding, material properties can be added as output nodes in the genotype (CPPN), resulting in a palette of materials which size will be the same as the voxels presented in the evolved structure. Another possible way of evolving these properties could be that two genotypes can represent the same individual, following two different encoding. Direct encoding can be used for the material properties, whereas, mutations and crossovers will then only held among the same genotype types.

Novelty search

Incorporating fitness information into the novelty search proved to be profitable for this diversity rewarding search adopting somehow some of fitness-based search advantages.

Fit individuals can be selected at the selection process resulting their survival and optimization during the evolution, as long as there will not be new fitter individuals.

As far as behaviors are concerned, a limited behavior space can benefit the search for novel individuals that their behavior belong only to this space, rewarding only those for their novelty. In this thesis, the space of behaviors was only normalized for trajectories of the robot bodies, whereas the orientation of their displacement played no role to the novelty search. A limited trajectory space could only take into consideration straight trajectories, treating all others as invalid behaviors and thus, not rewarding the individuals from which the trajectories were generated. It is expected that this type of novelty search will result in better solutions [12] as the diversity of locomotion patterns will only appeal to the strategy and not the direction. This technique used in [12], called *Minimal Criteria Novelty Search*, is a way of making the behavior space more compact so only "good" behavior will be rewarded for their novelty. Doing so, novelty search then incorporates indirectly more fitness information, which was not a point of interest during this thesis.

Chapter 7

Conclusion

point out superiority of novelty search over all setting almost

fitness based with competition better than pure

point novelty search with fitness elitism better than pure

Appendices

Appendix A

Simulation Settings

A.1 Environment

TABLE A.1: Voxelyze simulation settings

Property	Value	Description
DtFrac	0.9	The timestep of the simulation, currently 0.9 × *dt*, where *dt* is the optimal timestep.
ColSystem	3	Hierarchical collision detection between all voxels. Updates potential collision list only when aggregated motion requires it[1]
StopConditionValue	0.4	Time in seconds simulation is stopped.
TempBase	25.0	Base temperature of the environment.
TempAmp	39.0	Temperature's amplitude of the environment.
TempPeriod	0.025	Period of the temperature cycle.
Lattice_Dim	0.001	Lattice dimensions, each voxel has length, height, and depth of 1mm.

A.2 Materials

In this section all materials' properties used during the simulations will be given. All materials used in the simulations have a set of shared properties which are shown in table A.2. Furthermore, unique characteristics of the materials are presented in table A.4.

[1] From VoxCad's documentation [28].

TABLE A.2: Universal material properties

Property	**Value**	**Description**
Poisson's ratio	0.35	It is the ratio of expansion over two other axes following the compression in one.
Density	1×10^6 Kg/m^3	Density of material.
Temp phase	0	Phase of material to temperature period.
Static friction coef.	1	Static friction coefficient.
Dynamic friction coef.	0.5	Dynamic friction coefficient.

TABLE A.3: Unique per material properties

Name	**Color**	**Elastic Modulus** (MPa)	**CTE** ($1/deg\ C$)
Active positive (+)	Red	10	+0.01
Active negative (-)	Green	10	-0.01
Passive soft	Cyan	10	0.00
Passive hard	Blue	50	0.00

A.3 Experimental Settings

In this section the settings used for each experiment will be presented. For all the following experimental constants the simulation and material settings used are the ones described above, in case of other settings used, the new settings will be mentioned.

A.3.1 Settings

Objective function Displacement in body lengths (displacement divided by size of soft robot) of soft robot's center of mass.

Gravity acceleration $-27.6\ m/s^2$

Lattice dimensions $5 \times 5 \times 5$

A.3.2 Settings

Objective function Displacement in body lengths (displacement divided by size of soft robot) of soft robot's center of mass.

Gravity acceleration $-27.6\ m/s^2$

Lattice dimensions $10 \times 10 \times 10$

A.3.3 Settings

Objective function Displacement in body lengths (displacement divided by size of soft robot) of soft robot's center of mass.

Gravity acceleration $-27.6\ m/s^2$

Lattice dimensions $7 \times 7 \times 7$

A.4 Gravity Experiments

Objective function Displacement in body lengths (displacement divided by size of soft robot) of soft robot's center of mass.

Table A.4: Unique per material properties

Planet	**Dim.**	**Grav.** (m/s^2)	**Sim. Time** (Secs.)	**Temp. Period** (secs.)
Moon	7^3	-1.622	1.0	0.050
Mars	7^3	-3.711	1.0	0.050
Earth	7^3	-9.780	0.4	0.025
Jupiter	7^3	-24.790	0.4	0.025

Appendix B

Evolution Settings

Table B.1, presents the settings used in the evolutionary algorithm (CPPN-NEAT), the size of the population and the maximum number of generations are selected to match [40] for comparison purposes. The size of the competition used in the some experiment is 4. For novelty search, the nearest neighbor sparsity equation was used for the 10-closest neighbor behaviors, at the same time the threshold in all novelty search experiments used was tuned so ~ 0.8 behaviors per generation are generated.

Table B.1: CPPN-NEAT settings

Property	**Value**
PopulationSize	30.0
MaxGenerations	1000.0
DisjointCoefficient	2.0
ExcessCoefficient	2.0
WeightDifferenceCoefficient	1.0
FitnessCoefficient	0.0
CompatibilityThreshold	6.0
CompatibilityModifier	0.3
SpeciesSizeTarget	8.0
DropoffAge	15.0
AgeSignificance	1.0
SurvivalThreshold	0.2
MutateAddNodeProbability	0.03
MutateAddLinkProbability	0.05
MutateDemolishLinkProbability	0.00
MutateLinkWeightsProbability	0.8
MutateOnlyProbability	0.25
MutateLinkProbability	0.1
AllowAddNodeToRecurrentConnection	0.0
SmallestSpeciesSizeWithElitism	5.0
MutateSpeciesChampionProbability	0.0
MutationPower	2.5
AdultLinkAge	18.0
AllowRecurrentConnections	0.0
AllowSelfRecurrentConnections	0.0
ForceCopyGenerationChampion	1.0
LinkGeneMinimumWeightForPhentoype	0.0
GenerationDumpModulo	1.0
ExtraActivationFunctions	1.0
AddBiasToHiddenNodes	0.0
SignedActivation	1.0
ExtraActivationUpdates	9.0
OnlyGaussianHiddenNodes	0.0

Bibliography

[1] David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[2] Wikipedia. Evolutionary robotics — wikipedia the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Evolutionary_robotics&oldid=599436873`, 2014. Online; accessed 7-August-2014.

[3] Jonathan D Hiller and Hod Lipson. Evolving amorphous robots. In *ALIFE*, pages 717–724, 2010.

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

[5] Jason Gauci and Kenneth O. Stanley. A case study on the critical role of geometric regularity in machine learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008).*, Menlo Park, CA, 2008. AAAI Press. URL `http://eplex.cs.ucf.edu/publications.html#gauci.aaai08`.

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

[7] PicBreeder. `http://picbreeder.org/`. Online; accessed 7-August-2014.

[8] EndlessForms. `http://endlessforms.com/`. Online; accessed 7-August-2014.

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

[10] Kenneth Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

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

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

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

[14] Joel Lehman and Kenneth O Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336, 2008.

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

[16] Jean-Baptiste Mouret. Novelty-based multiobjectivization. In *New Horizons in Evolutionary Robotics*, pages 139–154. Springer, 2011.

[17] Jean-Baptiste Mouret and Jeff Clune. An algorithm to create phenotype-fitness maps. In *Proc. of the Artificial Life Conf*, pages 593–594, 2012.

[18] Deepak Trivedi, Christopher D Rahn, William M Kier, and Ian D Walker. Soft robotics: Biological inspiration, state of the art, and future research. *Applied Bionics and Biomechanics*, 5(3):99–117, 2008.

[19] Rolf Pfeifer, Max Lungarella, and Fumiya Iida. The challenges ahead for bio-inspired'soft'robotics. *Communications of the ACM*, 55(11):76–87, 2012.

[20] Filip Ilievski, Aaron D Mazzeo, Robert F Shepherd, Xin Chen, and George M Whitesides. Soft robotics for chemists. *Angewandte Chemie*, 123(8):1930–1935, 2011.

[21] Robert F Shepherd, Filip Ilievski, Wonjae Choi, Stephen A Morin, Adam A Stokes, Aaron D Mazzeo, Xin Chen, Michael Wang, and George M Whitesides. Multigait soft robot. *Proceedings of the National Academy of Sciences*, 108(51):20400–20403, 2011.

[22] Cecilia Laschi, Matteo Cianchetti, Barbara Mazzolai, Laura Margheri, Maurizio Follador, and Paolo Dario. Soft robot arm inspired by the octopus. *Advanced Robotics*, 26(7):709–727, 2012.

[23] Sangok Seok, Cagdas Denizel Onal, Robert Wood, Daniela Rus, and Sangbae Kim. Peristaltic locomotion with antagonistic actuators in soft robotics. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1228–1233. IEEE, 2010.

[24] Shigeo Hirose and Yoji Umetani. The development of soft gripper for the versatile robot hand. *Mechanism and machine theory*, 13(3):351–359, 1978.

[25] Jonathan D Hiller and Hod Lipson. Multi material topological optimization of structures and mechanisms. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1521–1528. ACM, 2009.

[26] Michael T Tolley, Robert F Shepherd, Bobak Mosadegh, Kevin C Galloway, Michael Wehner, Michael Karpelson, Robert J Wood, and George M Whitesides. A resilient, untethered soft robot. *Soft Robotics*.

[27] Siddharth Sanan, J Moidel, and CG Atkeson. A continuum approach to safe robots for physical human interaction. In *Int'l Symposium on Quality of Life Technology*, 2011.

[28] Jonathan Hiller and Hod Lipson. Dynamic simulation of soft heterogeneous objects. *arXiv preprint arXiv:1212.2845*, 2012.

[29] Stefano Nolfi, Dario Floreano, Orazio Miglino, and Francesco Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In *Artificial life IV: Proceedings of the 4th International Workshop on Artificial Life*, number LIS-CONF-1994-002, pages 190–197. MA: MIT Press, 1994.

[30] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 1994.

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

[32] Gregory S Hornby and Jordan B Pollack. Evolving l-systems to generate virtual creatures. *Computers & Graphics*, 25(6):1041–1048, 2001.

[33] Gregory S Hornby, Hod Lipson, and Jordan B Pollack. Generative representations for the automated design of modular physical robots. *Robotics and Automation, IEEE Transactions on*, 19(4):703–719, 2003.

[34] Jeff Clune, Benjamin E Beckmann, Charles Ofria, and Robert T Pennock. Evolving coordinated quadruped gaits with the hyperneat generative encoding. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 2764–2771. IEEE, 2009.

[35] Joshua E Auerbach and Josh C Bongard. Dynamic resolution in the co-evolution of morphology and control. In *Artificial Life XII: Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems*, number EPFL-CONF-191277, pages 451–458. MIT Press, 2010.

[36] Joshua E Auerbach and Josh C Bongard. Evolving cppns to grow three-dimensional physical structures. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 627–634. ACM, 2010.

[37] Jeff Clune and Hod Lipson. Evolving 3d objects with a generative encoding inspired by developmental biology. *ACM SIGEVOlution*, 5(4):2–12, 2011.

[38] Michał Joachimczak and Borys Wróbel. Co-evolution of morphology and control of soft-bodied multicellular animats. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, pages 561–568. ACM, 2012.

[39] Jonathan Hiller and Hod Lipson. Automatic design and manufacture of soft robots. *Robotics, IEEE Transactions on*, 28(2):457–466, 2012.

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

[41] Joel Lehman and Kenneth 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. ACM, 2011.

[42] Matthew Wall. Galib: A c++ library of genetic algorithm components. *Mechanical Engineering Department, Massachusetts Institute of Technology*, 87:54, 1996.