Evolving Scalable Soft Robots: Senior Thesis

Benjamin Berger

March 19, 2015

Abstract

Designing soft robots is difficult, time-consuming, and non-intuitive. Instead of requiring humans to engineer robots, this research uses genetic algorithms to evolve designs for robots that move when vibrated. Generative encodings are used to represent designs and are modified during the evolutionary process. A generative encoding is a set of rules that describe how to construct a 3D object. If the rules are applied over and over, it will create a larger and more complex robot. Current methods of evolving generative encodings fix a number of times to apply the rules *a priori*. This restricts the usefulness of the resulting encoding to one size. Instead, this research aims to create scalable solutions, with generative encodings capable of producing fit robots of various sizes. This is done by implementing the notion of a *pareto frontier* into the genetic algorithm. Each design produced is assigned to a category (small, medium, large, etc) and is judged on how well it can create robots that move when vibrated in each category. One design is deemed better than another if it can dominate across all categories. The resulting generative encodings should be able to produce soft robots of various sizes that can move when vibrated.

1 Introduction

Robots are becoming increasingly useful: automating physical tasks and performing beyond human capabilities of strength and precision. Robots allow us to operate in environments that can be hazardous for humans, such as performing search and rescue missions at a failing nuclear facility. Typical modern robots contain motors and are made of metal or are comprised of rigid structures. However, soft robots

can be made entirely out of plastic, rubber, silicon, or a variety of materials that allow them to be flexible and change shape [15]. For example, a snake-like soft robot can change shape to slither its way in between rubble of a collapsed building to look for potential survivors [9]. Some robots are being designed to grasp objects of unusual shape or of a delicate nature (like internal organs during surgery) [8, 1].

Designing soft robots is incredibly difficult. The structure of a robot is highly dependent on its purpose. If researchers designed a snake-like robot, the process of engineering a locomotion system would be time consuming, and the result strictly specific to that robot. If every soft robot had to be engineered by humans, the development process would be lengthy and its applications limited to a specific purpose.

Researchers are investigating automated methods for designing soft robots so that humans do not have to take the time to engineer them [12, 11, 3]. One such method is through genetic algorithms [12, 14, 3, 11]. The process works by using the same principles as Darwin's theory of natural selection: a population of soft robot designs is created, individual designs are tested for fitness, and successful designs are bred and mutated to create new designs. This process is depicted in Figure 1. To automate the selection process, researchers can test designs in computer simulation to avoid physically constructing every generated robot.

The tricky part of this process is representing designs in a way that genetic algorithms can utilize effectively. One method is through generative encodings, a set of rules that specify how a robot should be constructed. Each generative encoding represents a soft robot in a similar way that DNA can represent an organism. When an organism's DNA is changed, the traits that are expressed are changed. Similarly, when the rules of a generative encoding are altered, the resulting structure of the soft robot will be different. In this sense, the generative encoding is the genotype, while the resulting 3D model is the phenotype.

An interesting property of generative encodings is that they can create robots of various sizes from one ruleset. This is akin to building a brick wall. If you have a set of rules for laying bricks, and follow these rules for 10 bricks, you will build a small wall. Using the same rules, you can lay 100 bricks to build a larger wall, or 1000 bricks to build an even larger wall. Similarly with generative encodings, the more times the rules are applied, the larger and more complex the robot will grow. In this sense, one genotype can create numerous phenotypes. Figure 2 illustrates this principle well with a tetrahedral mesh.

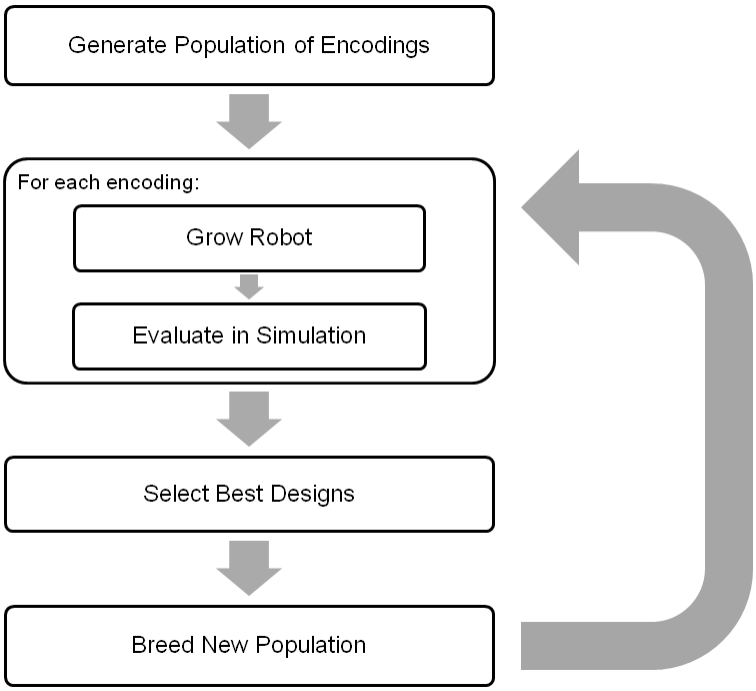

Figure 1: The genetic algorithm process this project uses to evolve generating encodings.

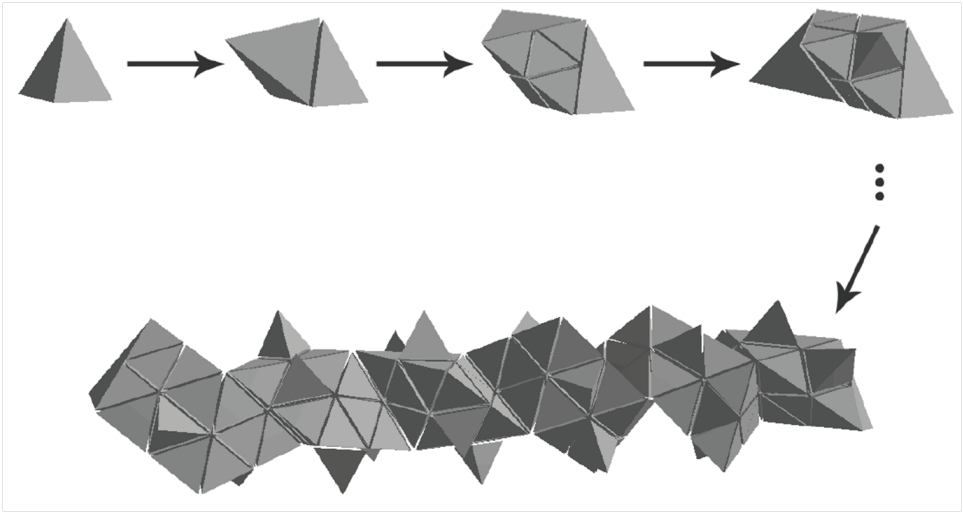

Figure 2: A tetrahedral mesh that is grown using a generative encoding. The robot grows a little larger and more complex each time the rules of the encoding are applied. One application of the ruleset can also be referred to as a "face rewrite". Picture from [13].

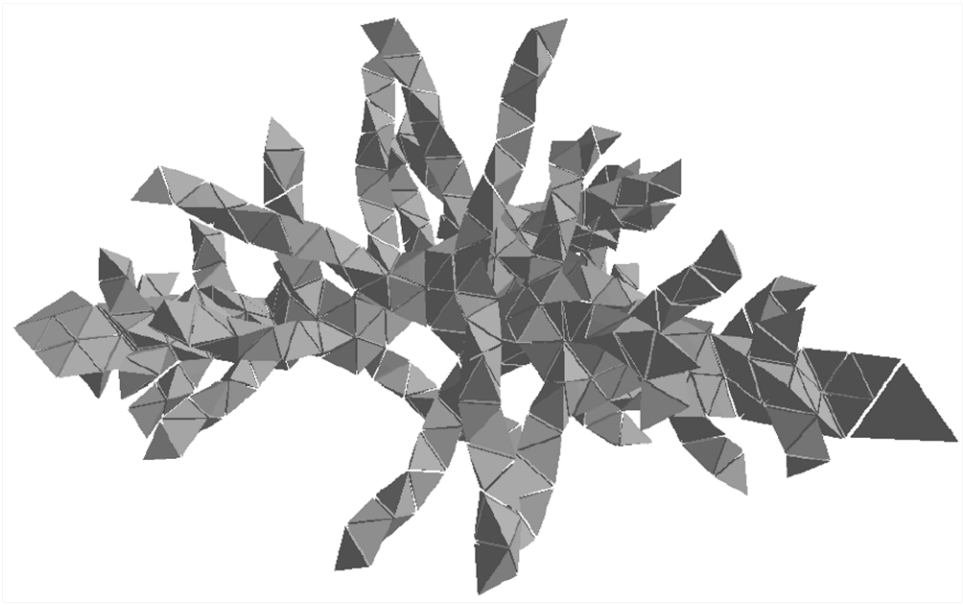

Figure 3: An example of a tetrahedral mesh grown using a generative encoding. Researchers can embed a pager motor into the structure to vibrate it, causing it to move. Picture from [13].

2 Background and Related Work

2.1 Generative Encodings

Generative encodings have existed for awhile in the computer science research community. Also known as formal grammars and L-systems, generative encodings have been used from natural language processing [6] to algorithmically generating plant structures [10]. Hornby and Pollack used generative encodings in combination with genetic algorithms to evolve tables (like the furniture) [7]. They showed that generative encodings produced better results at a faster rate than its non-generative counterpart.

2.2 Scalable Designs

Rieffel *et al*. used generative encodings to evolve tensegrity structures, comprised of tensile and rigid components, capable of deforming and changing shape [14]. One generative encoding can create tensigrity structures of various sizes. The more times the ruleset is applied, the larger the tensigrity grows. The gen-

erated structures exhibit patterns of regularity, repeating shapes as the size of the tensigrity increases. This type of scalability would be ideal to see from the generative encodings used to grow tetrahedral meshes in this research project. One generative encoding should be able to produce robots of various sizes that can move when vibrated.

Currently, many researchers evolve generative encodings to be of a specific size (fixed *a priori*). Using the brick wall analogy mentioned earlier, this would be like creating a ruleset that only works well if you lay 1000 bricks. If you use the same ruleset to lay 100 bricks, your wall is not guaranteed to be as good. Viswanathan and Pollack discussed how preselecting a fixed size of a generative encoding can retard evolutionary progress [16]. By not evaluating multiple phenotypes produced by the genotype, it can take more generations to achieve a specified level of fitness.

2.3 Project Foundation

Smith and Rieffel designed a generative encoding to represent soft robot designs like the one in Figure 3 [13]. They chose to represent the soft robots as tetrahedral meshes because it is the same way that physics engines (PhysX and Bullet) represent soft bodies in simulation. Additionally, STL files used for 3D printing represent structures as tetrahedral meshes, making it convenient to print and test robots in the real world. Each generative encoding is comprised of a different combination of the rules depicted in Figure 4.

Since soft robots are made of flexible materials, the structures naturally wiggle and jiggle. Conventional engineering of rigid structures tries to minimize vibration and mitigate the effect of resonance frequencies. For example, if a bridge vibrates and hits its resonance frequency, the structure can collapse (like the infamous Tacoma Narrows Bridge). For soft robots however, this research takes advantage of the material's natural flexible properties. Using a pager motor embedded in the robot, the structure is moved through vibration. The amount that the robot moves depends on its structure, and ultimately, the underlying generative encoding used to create the robot.

For his Senior thesis last year, Danise extended the work of Smith and Rieffel [13] by writing a program to simulate robots with the Bullet physics engine [3]. Danise also partially implemented a genetic algorithm. This work serves as the backbone of this research and is expanded upon in this thesis project.

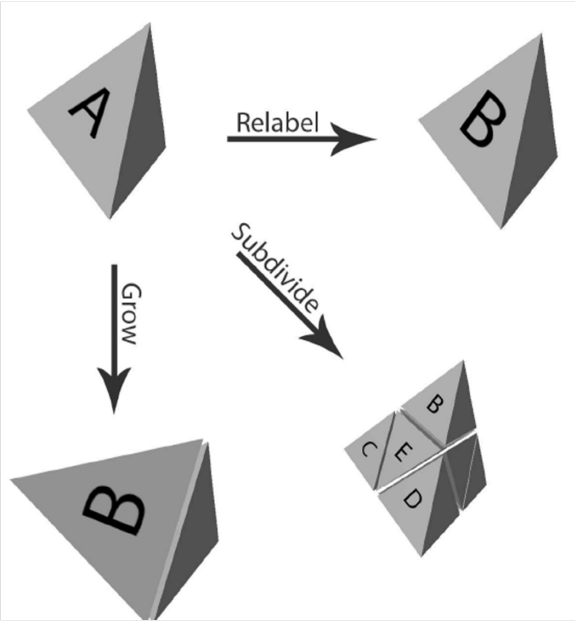

Figure 4: Example rules used in a generative encoding for tetrahedral meshes. Each open face of a tetrahedron is labeled and can have a rule applied to it according to what the generative encoding specifies. The process of evolving a generative encoding modifies the set of rules to produce new designs. Picture from [13].

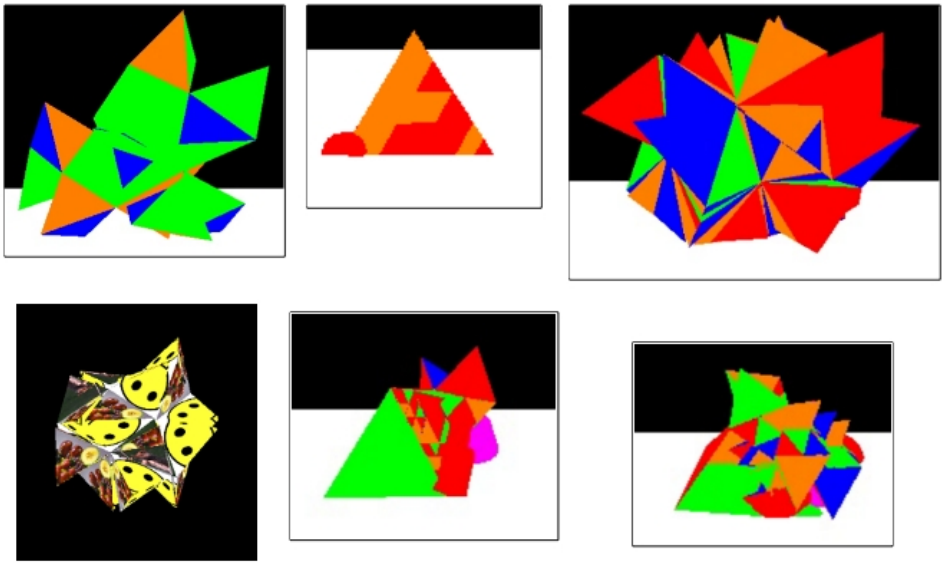

Figure 5: These are some robots that were created and tested in simulation by the genetic algorithm. Each individual is represented by a different generative encoding, each of which is comprised of a different combination of the rules depicted in Figure 4. Every robot is embedded with a vibrational mechanism, represented in the simulation as a pink mass that rotates around a red cylinder. The vibrational mechanism is most clearly visible for the middle robot on the bottom row.

3 Approach and Methods

The current methods used to evolve generative encodings do not create scalable solutions. Fixing a set number of times to apply a ruleset *a priori* creates soft robots that are only guaranteed to be fit for one size. This research alters the genetic algorithms used by Denise and Rieffel [3], aiming to create scalable generative encodings capable of producing soft robots of various sizes that can move when vibrated.

3.1 What is Scalable?

This thesis project alters the genetic algorithm process in order to evolve scalable encodings. Instead of evaluating a generative encoding by how far the robot it produces can move (with the level of growth preselected), all the robots the generative encoding produces are evaluated. Since there is no limit to how large a robot can grow, a stopping point is fixed arbitrarily. For example, if an upper limit of 200 face rewrites is chosen, all 200 possible robots a generative encoding can produce are simulated in order to properly evaluate scalability.

However, requiring that a generative encoding produce fit robots for every possible level of growth may not provide useful results quickly. For example, examine the caterpillar-like robot shown in Figure 2. A "leg" of the caterpillar shape may only appear after every 10th face rewrite. Having an underdeveloped leg may lower the fitness of the robot, meaning that only those with fully developed legs will be functional. In this scenario, the generative encoding used to produce the caterpillar-like robot is scalable, exhibiting patterns of regularity. Even though the generative encoding can produce unfit robots, it is still useful for every robot with fully developed legs. Although a generative encoding that can produce robots with fully developed legs after every stage of growth may be more desirable than one producing fit robots after every 10 face rewrites, researchers may decide the latter to be satisfactory. Further exploration to evolve the former may consume more time, computational resources, and money.

3.2 Changes to the Genetic Algorithm

Each robot simulated is evaluated by how far it can move when vibrated. Since each generative encoding can create multiple robots, generative encodings are compared with the concept of a *pareto frontier*. Robots are divided into categories of size (small, medium, large, etc) as shown in Figure 6. The best performing designs in each category are selected as representatives for their respective generative encodings. Generative encodings are then compared to each other with the graph shown in Figure 7. Generative encodings that outperform others across all categories are classified as dominating. After all individuals of a population are evaluated, the dominated individuals are removed and new encodings are created for the next generation.

The number of individuals chosen to be placed in each category is set as a parameter for the genetic algorithm. This number is referred to as the "category window size". The number of categories is also set as a parameter. If an experiment is run with the category window size set to 50 and the number of categories set to 3, the genetic algorithm will evaluate 50 small robots, 50 medium robots, and 50 large robots for each generative encoding.

4 Results and Discussion

4.1 Code Development

It was necessary to complete Danise's implementation of the genetic algorithm [3] in order to investigate this method of creating scalable soft robots. This involved restructuring the code base so that it could be extended with new features. One of these features was the ability to control the parameters of the genetic algorithm through command line arguments, allowing researchers to run experiments with different configurations without needing to recompile the program. The alterations and additions made to Danise's code enabled experiments to be run to completion, successfully evolving soft robots that can move when vibrated in simulation. The resulting generative encodings are not scalable; they are only guaranteed to produce fit robots grown to one size (the size that the genetic algorithm was configured to for the given experiment). Additionally, multiple experiments are able to be run in parallel on Union College's Jupiter

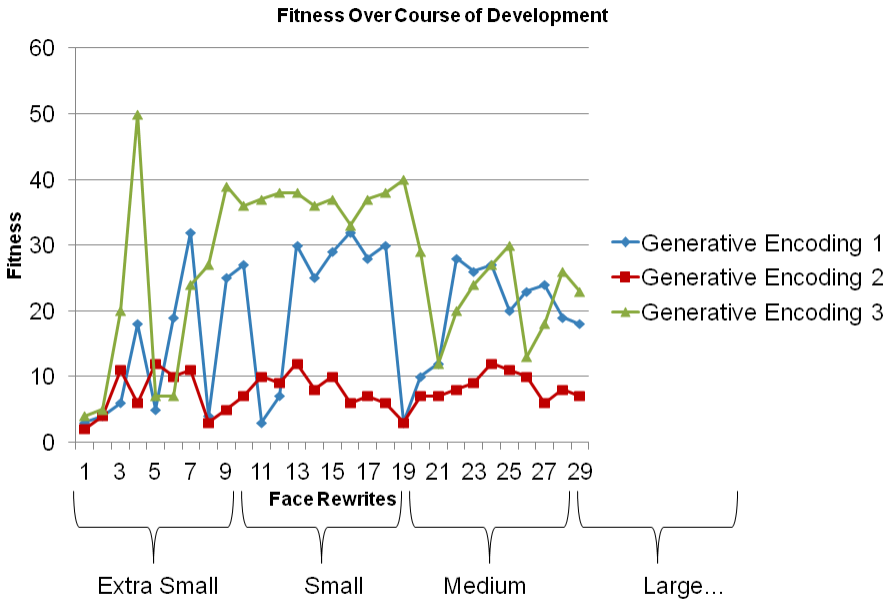

Figure 6: This figure shows the fitness over the course of development for three different generative encodings. Fitness over the course of development can also be referred to as an "ontogenetic trajectory". The robots produced by the encodings are separated into categories of size. Here, each encoding has 10 robots in each category.

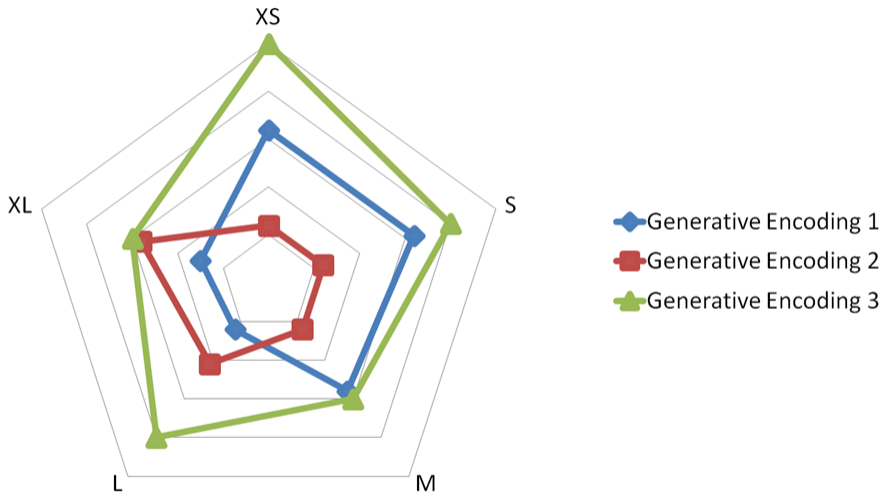

Figure 7: This graph was created using the data from Figure 6 in order to compare the best designs from each category. The further away a data point is from the center, the better the generative encoding performs in that category. As pictured in the graph, the data points for generative encodings 1 and 2 overlap, meaning that encoding 1 is better in some categories and worse in others; neither dominates the other. However, generative encoding 3 encircles both 1 and 2; it dominates across all categories.

Cluster, reducing the overall waiting time for collecting results.

4.2 Current Project Status

The code to evolve scalable soft robots is near completion. There are a few remaining issues that require attention, but they should be resolved soon.

Once the code is complete, one of the issues expected to be encountered is the emergence of a very large population size. There is a potential for an infinite number of non-dominated individuals to exist on the pareto front. Currently, only a minimum population size is maintained, but the population is allowed to grow larger if needed. If the number of non-dominated individuals is greater than or equal to the population size, the genetic algorithm is required to increase the population size and create new individuals for the next generation. For example, it can be specified that the genetic algorithm ensures at least 5 new individuals exist in every generation. If the population size is set to 20, but the population contains 18 non-dominated individuals, the genetic algorithm will create 5 new individuals, increasing the population size to 23.

4.3 Evaluation

The method of evolving scalable generative encodings here can be compared it to other evolutionary methods. As a baseline, it can be compared to Danise's non-scalable version [3]. To compare methods, two questions can be investigated:

1. Which evolutionary method can produce robots of a given fitness in less generations?
2. When an evolved generative encoding is given a random level of growth, how fit is the robot it produces?

Additionally, this method can be compared to a number of currently existing multiobjective algorithms [2, 18, 4, 5, 17]. These algorithms try to improve upon different aspects of the multiobjective search problem. For example PSEA-II [2] tries to promote diverse solutions by valuing individuals that are far from others in objective space.

4.4 Future Work

There are a number of different ways this research can be extended. For example, the robots produced by the genetic algorithm can be 3D printed and vibrated with a pager motor. Future researchers can experiment with different materials, or combinations of materials, and see if they can accurately model these materials in simulation. Future researchers could also change the frequency of the vibrational mechanism, or create a new generative encoding that utilizes multiple motors.

Additionally, researchers can investigate the application of this genetic algorithm process in other contexts. If the soft robot design problem is changed out for a different multiobjective problem, how well does this method compare to existing algorithms? What affect will varying the population size, or the category window size, have on the performance? If the number of categories is increased in the middle of an experiment, similar to Rieffel [11], will it produce comparable results while cutting computation time?

5 Conclusion

This research aims to create scalable designs for soft robots that can move when vibrated. Extending the work of Danise and Rieffel [3] successfully evolves generative encodings to produce robots that can move when vibrated. However, these robots are only fit when grown to a specific size; the encodings are not scalable. This thesis outlines a method for evolving scalable solutions. The code is currently under development and will be completed soon. Afterwards, this method of creating scalable designs will be compared to other evolutionary and multiobjective algorithms.

References

[1] Eric Brown, Nicholas Rodenberg, John Amend, Annan Mozeika, Erik Steltz, Mitchell R Zakin, Hod Lipson, and Heinrich M Jaeger. Universal robotic gripper based on the jamming of granular material. *Proceedings of the National Academy of Sciences*, 107(44):18809–18814, 2010.

[2] David W Corne, Nick R Jerram, Joshua D Knowles, Martin J Oates, et al. Pesa-ii: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2001*. Citeseer, 2001.

[3] Andrew Danise. Evolving soft robots with vibration based movement, 2014.

[4] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[5] Carlos M Fonseca and Peter J Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, 3(1):1–16, 1995.

[6] Eli Goldberg, Norbert Driedger, and Richard I Kittredge. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53, 1994.

[7] Gregory S Hornby and Jordan B Pollack. The advantages of generative grammatical encodings for physical design. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 600–607. IEEE, 2001.

[8] 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.

[9] Ming Luo, Weijia Tao, Fuchen Chen, Tri K Khuu, Selim Ozel, and Cagdas D Onal. Design improvements and dynamic characterization on fluidic elastomer actuators for a soft robotic snake. In *2014 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, 2014.

[10] Przemyslaw Prusinkiewicz, Aristid Lindenmayer, and James Hanan. The algorithmic beauty of plants. *The virtual laboratory (USA)*, 1990.

[11] John Rieffel. Heterochronic scaling of developmental durations in evolved soft robots. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, pages 743–750. ACM, 2013.

[12] John Rieffel, Davis Knox, Schuyler Smith, and Barry Trimmer. Growing and evolving soft robots. *Artificial life*, 20(1):143–162, 2014.

[13] John Rieffel and Schuyler Smith. A face-encoding grammar for the generation of tetrahedral-mesh soft bodies. In *ALIFE*, pages 414–420, 2010.

[14] John Rieffel, Francisco Valero-Cuevas, and Hod Lipson. Automated discovery and optimization of large irregular tensegrity structures. *Computers & Structures*, 87(5):368–379, 2009.

[15] 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.

[16] Shivakumar Viswanathan and Jordan Pollack. How artificial ontogenies can retard evolution. In *Proceedings of the 2005 workshops on Genetic and evolutionary computation*, pages 273–280. ACM, 2005.

[17] Eckart Zitzler, Marco Laumanns, Lothar Thiele, Eckart Zitzler, Eckart Zitzler, Lothar Thiele, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm, 2001.

[18] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *evolutionary computation, IEEE transactions on*, 3(4):257–271, 1999.