# Scalable Co-Evolution of Soft Robot Properties and Gaits

By

Davis K. Knox

*********

Submitted in partial fulfillment

of the requirements for

Honors in the Department of Computer Science

UNION COLLEGE

June, 2011

# Scalable Co-Evolution of Soft Robot Properties and Gaits

John Rieffel
Computer Science Department
Union College
Schenectady, New York 12308
rieffelj@union.edu

Davis Knox
Reamer Campus Center Box 1114
Union College
Schenectady, New York 12308
knoxd@union.edu

## ABSTRACT

The field of soft robotics is very promising; applications include urban search and rescue and covert surveillance, but these projects are not yet realized, partly because of the difficulties in soft robot shape and locomotion design. Because of this, traditional design methods do not prove to be effective. This project attempts to come up with solutions to this soft robot design problem; utilizing a genetic algorithm, a computer simulation of Darwin's "Survival of the Fittest," this project attempts to make soft bodies move. This genetic algorithm evaluates each solution in simulation, and assigns each one a fitness based on distance travelled. Furthermore, this project implements a technique called co-evolution, which evolves two different things in lockstep, utilizing new found advancements in one to help bolster the other. This project evolves soft bodies' physical properties, values that affect how they move, alongside locomotion techniques, the gaits defining their movement. Optimizations to this process are realized in the use of scalable soft meshes; this system starts on a simple mesh, and slowly increases its density, reducing the overall computation time.

## 1. INTRODUCTION

The field of Soft Robotics hopes to go places that conventional rigid robotics has not gone. Imagine a robot able to traverse rubble, squeeze through small cracks and crevices, and expand and contract. Unfortunately, soft robotics is still in its infancy; almost all projects are not yet realized, partly because of the complexities associated with having all soft parts. Because of these complexities, traditional design methods for both morphology and locomotion design for soft robots does not prove to be effective [8]. In order to tackle this hard design problem, it would be beneficial to turn to unorthodox methods of design.

Previous work has shown that genetic algorithms are able to approach hard design problems; in 2001, Pollack utilized a co-evolutionary algorithm to design robots and their controllers [5]. Also in 2001, an L-system driven genetic algorithm was used to produce virtual creatures [3], and a generative genetic algorithm was used to produce table like structures [2]. More recently, genetic algorithms were applied to the design of modular robots [4], and complex tensegrity structures [8].

All of these projects show genetic algorithms' ability to approach hard design problems, but they all employed genetic algorithms in the physical design of these structures. This work hopes that the genetic algorithm's ability to approach hard problems can be applied in an equally difficult design realm in soft robotics: locomotion.

While physically building soft robots and testing locomotion techniques in the real world would be the ultimate goal of this research, we instead turn to computer simulation in order to evaluate many possible solutions quickly and easily. Until recently, computer simulation of soft bodies was not supported by physics engines, but now, the newly released version of NVidia's PhysX video game physics engine adds the functionality to simulate soft shapes. Previous work has shown that PhysX is indeed capable of being harnessed by a genetic system [7]. Our process utilizes PhysX in the simulation of possible locomotion techniques for our soft robot.

As we show, the system is able to produce novel solutions to the soft robot locomotion problem. These solutions produce what conventional thought could not, to make a soft robot move.

## 2. THE PROJECT

This project employs the use of a co-evolutionary genetic algorithm to evolve soft robot physical properties alongside gaits on a static morphology in order to try to find locomotion strategies. The gaits are defined by a set of muscles and their firing patterns. Muscle placement is set by the experimenter before evolution, but may be changed between runs in order to try out different placements. The firing patterns for the muscles are evolved by the system. The physics engine PhysX is utilized to simulate the soft bodies' locomotion techniques. Inside of PhysX, the soft body is represented by a soft mesh; this is a tetrahedron mesh constructed to represent the passed model file. The simulation is bottlenecked by the density of the soft mesh; when the mesh has few vertices, the simulation runs quickly, but if the soft mesh has many vertices, the simulation runs much more slowly. To speed up computation, the system starts evolution on a low density mesh, and increases its density

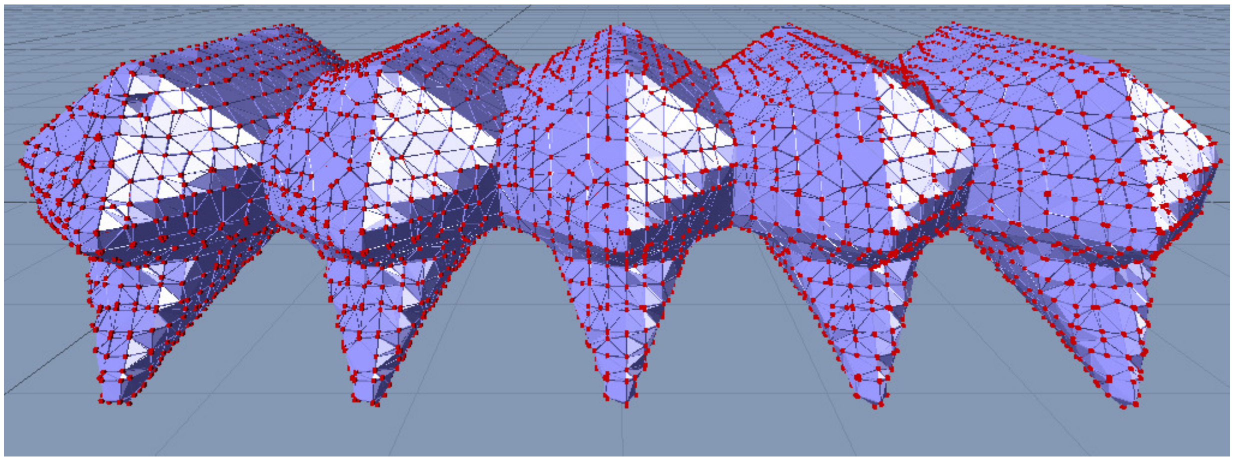

**Figure 1: This is an example of a complex soft mesh, or one with very high vertex density.**

over time.

## 2.1 Co-Evolutionary Genetic Algorithms

To understand what a co-evolutionary genetic algorithm is, it is necessary to first understand a genetic algorithm. A genetic algorithm can be described as a computer simulation of Darwin's "Survival of the Fittest". The algorithm starts out with a randomly produced starting population of a size set by the experimenter. Then, all members of the population are evaluated by some fitness function. After every member of the population has a fitness, it is sorted and the weaker members of the population are culled. To refill the population, the algorithm utilizes mutation, changing one aspect of a solution, and crossover, taking parts of two parents and combining them into one solution, on the remaining fit members of the population. The process then starts again, with another cycle of evaluation, sorting, culling, and repopulation. Over many iterations of this cycle, the solutions produce better and better results. The process ends when a plateau in fitness is reached or when stopped by the experimenter [1].

A co-evolutionary genetic algorithm follows much of the same structure as a genetic algorithm, but instead of only evolving one thing it evolves two. Co-evolutionary genetic algorithms are employed when there are multiple things defining the end result of evaluation. Because evolving both things at the same time would introduce too many variables into the system, it instead evolves one at a time. A co-evolutionary algorithm can then be thought of residing in two phases. The first is evolving one of its qualities for a set number of generations while the other remains static. Then, it takes its best solution at that point, and plugs it into the evolution of the other defining quality. Using this method, a co-evolutionary genetic algorithm produces results where each defining quality is shaped by the other, and therefore overall fitness is hopefully higher than if they were evolved separately.

## 2.2 Scalable Soft Meshes

As previously stated, this project uses NVidia's PhysX physics engine in order to evaluate the locomotion techniques of the soft bodies in simulation. PhysX sees the soft bodies as tetrahedron meshes. These meshes are made up of some number of vertices and can be simple or complex in their vertex density. In the real world, a soft body would have infinite "vertices", but in simulation this is not the case. The fewer vertices a mesh has, the faster it will run in simulation. The downside of this is that because of the low vertex count, the mesh is not a very realistic representation of our soft body. Therefore, if a mesh is dense in vertices, while it is much closer to a realistic representation of our soft body, the simulation runs much slower. This system allows the density of the mesh to be changed over the course of an experiment, starting on a simple mesh and increasing its density over time, decreasing the overall computation time.

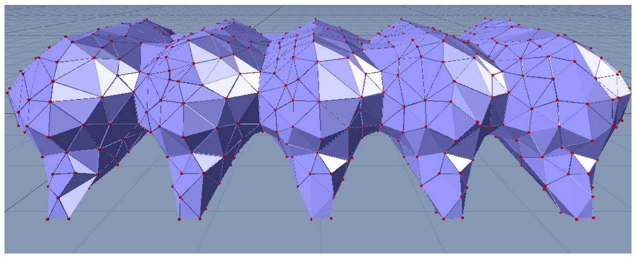

**Figure 2: This is an example of a simple soft mesh, or one with a low vertex density.**

## 2.3 Morphologies and Setting Muscles

The morphology of the soft body as well as the muscle placements on the morphology are static during evolution. So where do these things come from? The soft body shape that this system uses was engineered by hand, and the muscle placement is set by the experimenter.

To do this, the experimenter brings the hand engineered model into a modified version of PhysX viewer. This application allows the experimenter to set the mesh to the desired vertex density and add muscles where desired. At any time, this application can export an .obj and .tet file which can be imported into our system where the evolution takes place.

The experimenter is able to add a muscle between any two vertices in the soft mesh, as seen in figure 3. Since vertex

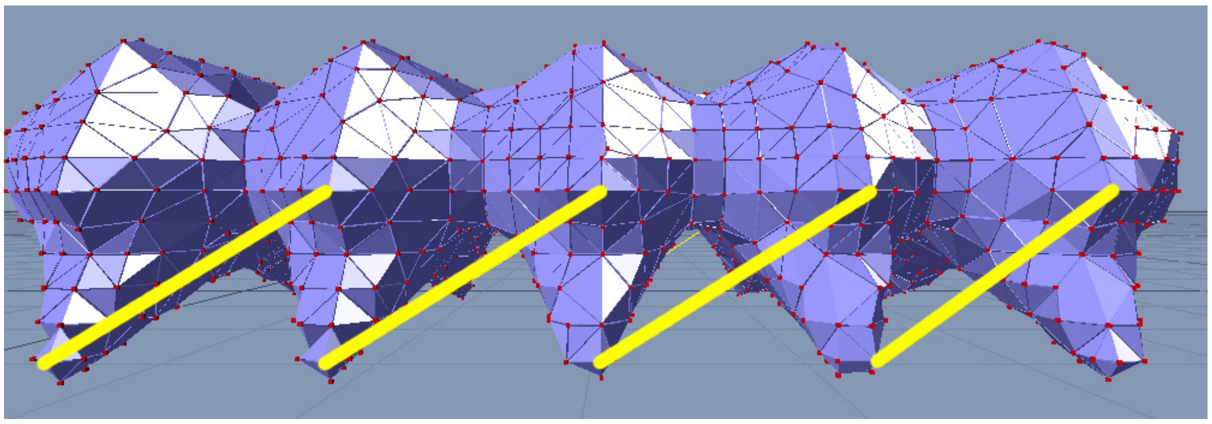

**Figure 3: This is an example of a soft mesh in our modified PhysX viewer application. The yellow lines between vertices represent muscles that the experimenter has added to the soft mesh. They have also been manually bolded to make them easier to see. When exported, an .obj and .tet file are created with the soft mesh, along with a .txt file that keeps track of muscle placement.**

position on the mesh varies between mesh densities, when a muscle is added to the soft mesh, the positions of the two end point vertices are stored. Then, when the experimenter makes the mesh more dense, the muscle is updated by searching through all vertices, selecting the ones closest to the previous end point positions.

This application also allows for easy addition of symmetrical muscles to the soft mesh. The experimenter first adds muscles to one side of the soft mesh. Then, with the simple press of a button, the program adds symmetrical muscles to the other side of the soft mesh. This process involves taking the positions of the two muscle vertices, translating to the opposite side of the soft mesh, and ray casting back into the soft mesh to find opposite positions on the soft mesh. Then, a simple search is performed to find the closest vertices to these positions, creating a symmetrical muscle. This process is then repeated for all muscles, finding symmetrical vertices for them all.

## 2.4 The System

The problem of locomotion in soft robotics is a chicken and egg problem. In order to design gaits for the soft robot, the physical properties of the soft robot material must be known so that the gaits can be tailored to how the soft body behaves in the world. On the other hand, in order to choose the physical properties of the soft body material, the types of gaits must be known so that properties can be chosen to maximize the potential of the gaits.

The system that performs the work of coming up with locomotion techniques utilizes a co-evolutionary genetic algorithm that evolves physical properties along side gaits. In order to approach this problem, the system starts with giving a “best guess” for the physical properties, and sets the firing pattern of each muscle to random. Then, through the evolution of these qualities, both physical properties and gaits can be refined to get more and more distance travelled.

To evaluate how fit each solution actually is, it is dropped into simulation with the given physical properties and mus-

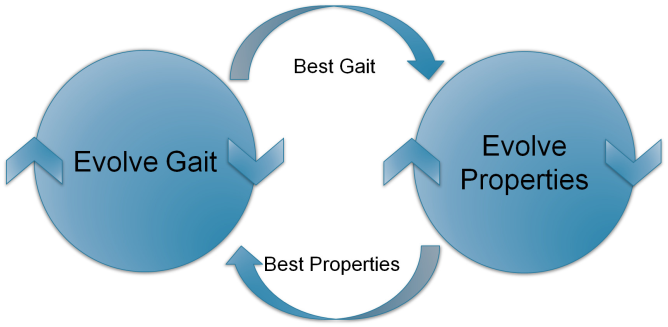

**Figure 4: This is a diagram of the co-evolutionary system; gaits are evolved using static physical properties. Then the best gait is made static, and the system evolves physical properties. This cycle continues until stopped by the experimenter.**

cle firing patterns and left to simulate for eight thousand time steps. After these eight thousand time steps, the total distance that the soft body moved is recorded and assigned as that solution’s fitness. This process is repeated for each solution in the population, which there are forty. Every ten generations, the system switches from evolving gaits to evolving physical properties or from evolving physical properties to evolving gaits. Upon switching phases, the system takes the current best solution of that phase and sets it as static in the evolution of the other phase.

The point at which the mesh becomes more dense is set by the experimenter. During the experiment, the mesh can be made more dense as often as the experimenter desires, as long as they have exported the corresponding meshes from the modified PhysX viewer application as discussed in the previous section.

### 2.4.1 *Defining Gaits*

In the system, a gait represents a set of firing patterns for the muscles on the soft body. A firing pattern is defined by three separate variables: duty cycle, phase, and period. Duty

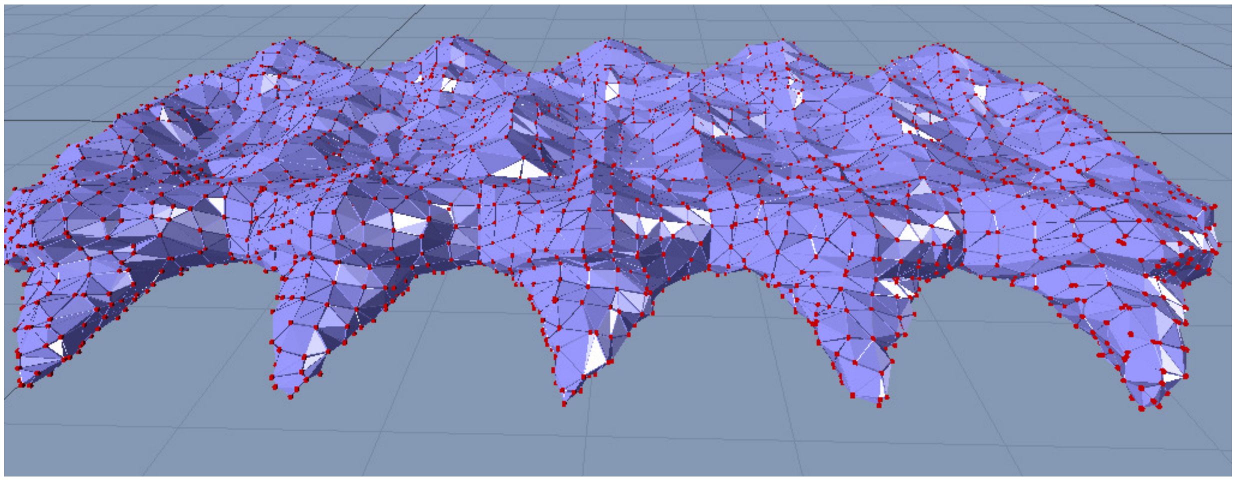

**Figure 5: This is an example of a soft mesh that has "limp" physical properties. More specifically, this soft mesh would have low stretching stiffness, because it is not holding its shape very well. Also, it would still have relatively high volume stiffness, because is still retains much of its original volume.**

cycle represents the percent of time that a muscle is being pulled during its period. The period of the firing pattern represents the time between rising edges in the muscle's duty cycle. Finally, the phase of the firing pattern represents the delay between from when the simulation's period starts and the muscle starts its own period cycle.

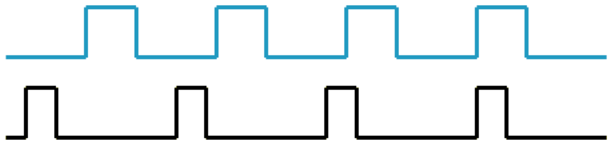

**Figure 6: These two firing patterns define when their respective muscles are being pulled. When the line rises, the muscle contracts, and when the line falls, the muscle is released. A gait is made up of many firing patterns, one for each muscle.**

### 2.4.2 *Physical Properties*

The physical properties that this system is able to manipulate is limited by what PhysX allows the experimenter to set. Of the set of possible properties, this system manipulates four: stretching stiffness, volume stiffness, damping coefficient, and friction.

- **Stretching stiffness** dictates how much a soft body attempts to hold its starting shape. Therefore, a soft body with very high stretching stiffness will try its hardest to maintain its shape, while a soft body with very low stretching stiffness will flop into a pool on the ground.
- **Volume stiffness** dictates how much a soft body attempts to hold its starting volume. While this does not have as drastic an effect as stretching stiffness on the soft body's behavior in simulation, it does still change how the soft body interacts with the world. For instance, a soft body with very low stretching stiffness will pool onto the ground, but if the volume stiffness remains high, the soft body will still be three dimensional. If volume stiffness is very low alongside very low stretching stiffness, the soft body will become a two dimensional pool on the ground, like a liquid.

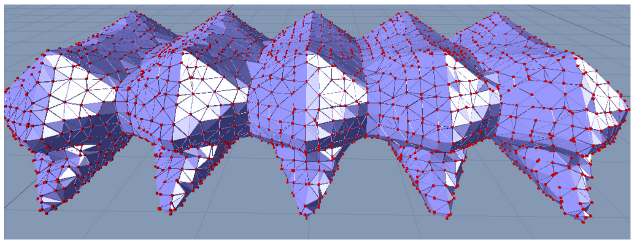

**Figure 7: This soft mesh has "stiff" physical properties. More specifically, this soft mesh has high stretching and volume stiffness, because it maintains its volume and shape.**

- **The damping coefficient** of a soft body changes how fast it returns to equilibrium after moving in the world. With a very high damping coefficient, if a soft body is pulled it will oscillate very little before returning to equilibrium. With a very low damping coefficient, if a soft body is pulled it will oscillate at its natural frequency defined by stretching and volume stiffness before returning to equilibrium.
- **The friction** of the soft body dictates the amount of friction that the soft body has with the ground in simulation. If friction is zero, if the soft body can get moving it will not stop, while if the friction is maximum, the soft body will be affected by full friction if it starts to move.

Some bounds were put on all of these properties in the system. If stretching stiffness is let to be all possible values, half of the time the soft body is a pool on the ground incapable of utilizing the evolving gaits. Likewise, if friction is

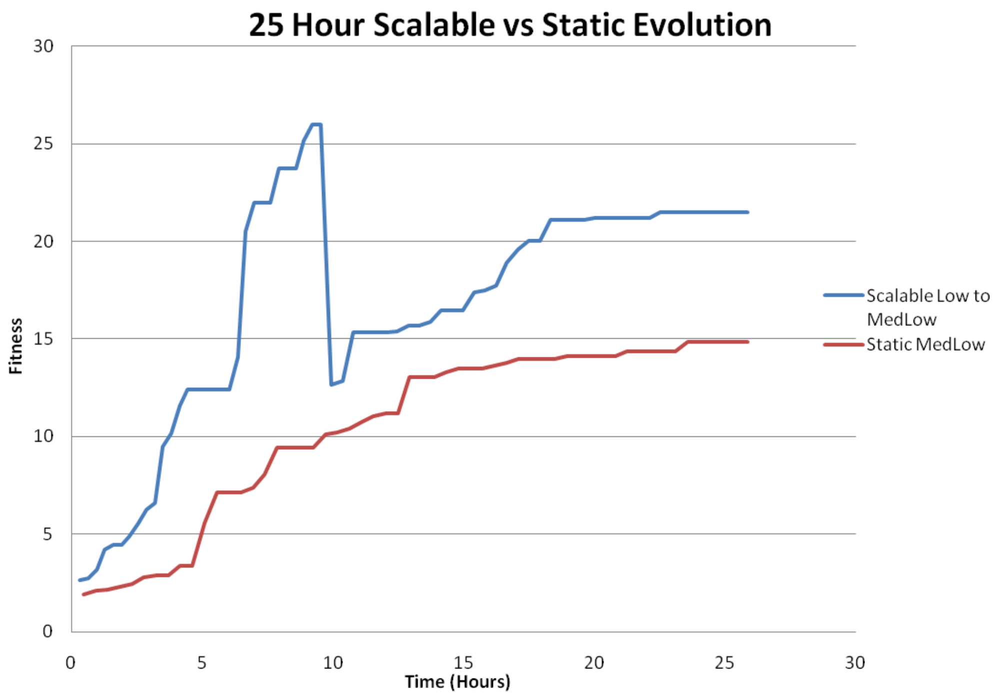

**Figure 8: Graphed data from two runs, one static and one scalable. The blue line shows the fitness over time of a scalable run from the low mesh to the medium low mesh. The large dip in fitness around the ten hour mark occurs when the mesh switches from low to medium low.**

allowed to be too low, the walking patterns of the soft body have no affect because it cannot push off of the ground in simulation.

## 2.5 Results

The system was able to produce novel solutions to the soft robot locomotion problem. In some cases, the scalable mesh run performed much better than the corresponding static mesh run. However, in other cases, the scalable mesh run did not perform significantly better than the corresponding static mesh run, and in some cases, the static run actually performed slightly better than the scalable run.

**Table 1: Experimental Results**

| Fitness | Hours | End Mesh | Mesh Switches |
|---|---|---|---|
| 40.16 | 34 | Low | 0 |
| 46.94 | 42 | Low | 0 |
| 24.61 | 72 | Max | 0 |
| 14.86 | 25 | MedLow | 0 |
| 21.47 | 25 | MedLow | 1 |
| 11.97 | 24 | Med | 0 |
| 12.86 | 24 | Med | 1 |
| 15.99 | 48 | Med | 0 |
| 16.65 | 48 | Med | 1 |
| 15.91 | 48 | Med | 2 |
| 6.18 | 46 | Max | 4 |

In the experiments run during this work, there were five meshes available to the experimenter: low, medium low, med, high, and maximum. All of the runs listed above except for the static runs that did not start on the low mesh, started on the low mesh. Therefore, runs that are listed with a mesh switch count of one, changed from the low mesh to their end mesh. Runs listed with a mesh switch counter greater than one ran on an intermediate mesh(s) before reaching their end mesh.

The limitation of the scalable mesh system is seen in the large loss of fitness between the different meshes, as observed in figures 8 and 9. This is because when the soft meshes are augmented to contain more vertices, it fundamentally changes how they act in simulation. Furthermore, the physical properties and gaits specifically tailored to those meshes do not transfer as well as hoped when switching from mesh to mesh. Still, it is important to realize that while the benefits of using scalable meshes in the evolution process is not as significant as hoped, there is still enough transfer of fitness so that the scalable system still does well and out performs static evolution most of the time.

One outlier from the rest of the data is the last entry in the table; the scalable run that went through all of the meshes from low to max. After forty six hours of evolution, the end result only had a fitness of 6.18 units travelled in the PhysX simulation. This can be explained by the fact that the run changed meshes four times. Because of all of these mesh switches, and the fact that each time the mesh is switched there is a large dip in fitness, the scalable mesh feature turned from a benefit to a detriment.

We can see this trend continue in the data visualized in figure 9. After the end of all the runs, the scalable run with one mesh switch performed the best, with a fitness of 16.65

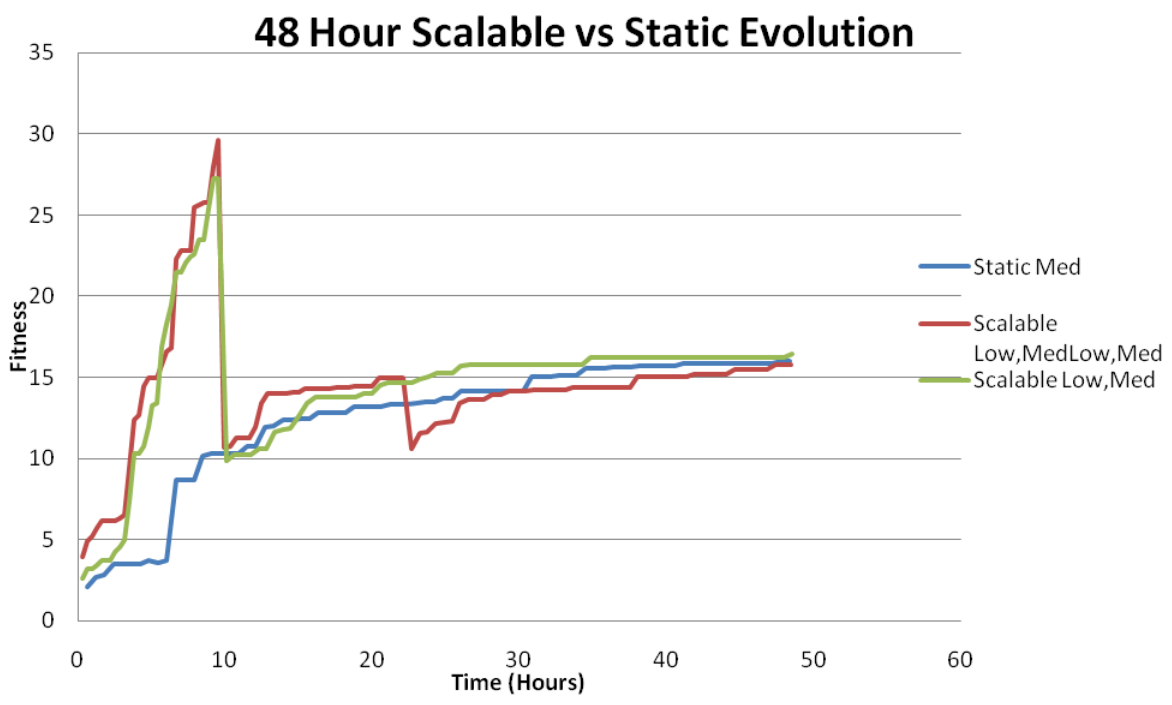

**Figure 9: Graphed data from three runs. The blue line shows a static medium mesh run. The red line shows a scalable run with two mesh switches, one around the ten hour mark, and one around the twenty two hour mark. The green line shows a scalable run with one mesh switch around the ten hour mark. At the end of the runs, all of them have a similar fitness, but the one switch scalable run is the highest, followed by the static run, and lastly the two switch scalable run.**

units travelled. In second place was the static run, with a fitness of 15.99, and lastly was the scalable run with two mesh switches, which had a fitness of 15.91 units travelled.

It is also important to point out that a genetic algorithm is still a form of random search. During each evolution, the possibility of finding a much better solution is mostly chance based; in figure 9, the static run had a very large increase in fitness at around the six hour mark. Before and after this large fitness increase, the static run increased its fitness very slowly. If this breakthrough had not happened during the six hour mark, the end result of the run might show both scalable runs largely outperforming the static run; at the end of the day, this system is still playing the evolution lottery.

## 2.6 Difficulties

PhysX is a complex system with little documentation available. Additionally, the soft body features of PhysX are extremely difficult to find documentation on because they are technically still in BETA. This being said, modifying PhysX viewer was a hard task to accomplish. It was essentially written as a closed product, and modifying the code to do the required tasks meant jumping through seemingly endless hoops and constantly tracing calls line by line with debugger software. Eventually, after the modifications to PhysX viewer were complete, the largest problem with this project reared its ugly head.

Since the soft body features of PhysX are still in beta, all functionality is not completely bug free. A problem arose with one of the most essential parts of the system's code, pulling a muscle. The code to pull a muscle entails applying equal and opposite forces between the two vertices of a muscle, but in addition to simulating a muscle pull, this also causes the soft body to spin around like a top. NVidia was contacted about the bug and they confirmed that they could reproduce it, but unfortunately a fix is not available at this time. Hopefully when the soft body features of PhysX leave BETA stage, this bug will no longer exist.

In order to still be able to run our system with this bug affecting simulation, a work around was implemented. An equal and opposite force was applied to a vertex on each end of the soft body along an axis in simulation every time step. This constant force locked the soft body to stay aligned with the axis. While this work around let the system evolve gaits that moved the soft body in a straight line, it also limits the genetic algorithm's search space, which is a bad thing. One of the best qualities of genetic algorithms is their ability to produce unexpected working results. It could be that a effective gait for our soft body would be to walk in circles, spinning in a line, but this work around inhibits these non linear gaits from being discovered.

Another problem with the system was the affect that scalable meshes had on the pulling of muscles. If an equal force is applied to pulling a muscle on different mesh densities, the mesh with the higher vertex density will qualitatively pull much harder than the lower density mesh. No documentation exists on this occurrence, and so to get muscles on different meshes to pull with the same strength, the experimenter was required to go into simulation and slowly walk through the meshes, slighting tweaking and re-tweaking the pull strengths of the muscles until they were are qualitatively equal across the meshes.

### 2.7 Discussion

The genetic algorithm was very good at finding weaknesses with the system. Although the initial hope of utilizing scalable meshes as a way to greatly reduce computation time allowed the scalable system to outperform a static system in some experiments, it is clear that full utilization of this concept is limited. The types of gaits that the system produced varied depending on what mesh the evolution started on. For instance, all evolutions that started on the "low" mesh produced bi-pedal gaits because bi-pedal gaits were very effective on the low mesh. Conversely, gaits produced from the "maximum" mesh produced wave like gaits that initiated a wave movement on one end of the soft mesh and propagated it though the soft mesh in order to propel it forward. While wave-like gaits worked well on the "maximum" mesh, the bi-pedal gaits produced on the lower meshes did not translate well to the "maximum" mesh. Therefore, the starting mesh for an experiment must be chosen in relation to which mesh the experiment will end on. This limits the differences in meshes available to the experimenter, and also the large computation speed differences that go along with them.

In addition, the number of mesh switches is limited by this loss in fitness between different densities. Compared to static evolution, the scalable system seemed to do better when there were as few switches as possible, namely one. In these cases, the scalable mesh system acted as a "spring board" to higher fitness, as the majority of the computation time still took place on the final mesh.

Because of these points, it is unclear if this method will ever be completely viable for real world applications. If the soft bodies' locomotion fitness drops so substantially between an increase of only a few thousand vertices, the drop in fitness could be even higher between an extremely high resolution mesh and the real world, where the soft body would essentially have infinite "vertices".

On the other hand, it could be that some features of the current system are not working as expected, and are causing this large decrease in fitness between mesh densities. Further exploration of the current system and its enclosed variables are necessary in order to pin the blame for these large fitness drops solely on the use of scalable meshes.

Along the same lines, because there is no mathematical relationship between muscle pull strengths, and also because of the fact that they were balanced by hand, there could be differences in the pull strengths between the different meshes. These pull differences could be a factor in the fitness loss between different density meshes.

## 3. CONCLUSIONS

The scalable co-evolutionary system was able to produce novel solutions for the soft robot locomotion problem. The system evolved soft body physical properties alongside gaits on a static morphology to reach these fit locomotion techniques. Some optimizations to this process were realized in the use of scalable soft body meshes, but their use was limited by the fact that a loss of fitness was seen when a switch of mesh occurred.

## 4. FUTURE WORK

It is worth noting that this scalable mesh process is not a refined one. It is very possible that by the tweaking some of the variables in the current system like when to switch from mesh to mesh, how long to run experiments, which meshes to start and end on, and how different the meshes are, results could be produced which dwarf the ones shown in this paper.

Eventually, after this work is complete, the next step would be to use genetic algorithms to not only design the locomotion techniques for soft robots, but also to design their morphologies as well. A generative grammar like the one proposed by Rieffel and Smith could power a genetic algorithm to create new soft tetrahedral meshes for soft robot morphologies [6].

## 5. REFERENCES

[1] J. H. Holland. Genetic algorithms. *Scientific American*, 1992.

[2] G. S. Hornby and J. B. Pollack. The advantages of generative grammatical encodings for physical design. In *Congress on Evolutionary Computation*, pages 600–607, 2001.

[3] G. S. Hornby and J. B. Pollack. Evolving l-system to generate virtual creatures. *Computers and Graphics*, 25(6):1041–1048, 2001.

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

[5] J. B. Pollack. Three generations of automatically designed robots. *Artificial Life*, 7(3):215–223, 2001.

[6] J. Rieffel and S. Smith. A face-encoding grammar for the generation of tetrahedral-mesh soft bodies. pages 414–420, 2010.

[7] S.-F. N. S. Z. H. H. S. R. J. Rieffel, J. and B. Trimmer. Evovling soft robotic locomotion in physx. 2009.

[8] V.-C. F. Rieffel, J. and H. Lipson. Automated discovery and optimization of large irregular tensegrity structures. *Computers and Structures*, 87(5-6):368–379, 2009.