Evolutionary Design of Mechanical Linkages

Amanda Ghassaei
Center for Bits and Atoms
Massachusetts Institute of Technology
ghassaei@mit.edu

Joy Ming
School of Engineering and Applied Sciences
Harvard University
jming@college.harvard.edu

ABSTRACT

Mechanical linkages, or systems of rigid bars connected by hinges or other rotating components, have applications spanning the fields of robotics, animation, biology, art, and design. However, designing linkages with desired motion is a highly unintuitive process, involving rigorous experimentation in a high dimensional parameter space. We propose to use the principles of Darwinian evolution to perform computational search and optimization of 2D planar linkages. We do so in two types of contexts: curve-fitting and task-based fitness. In this paper we describe the creation of a user interface for linkage optimization, our operationalization of the problem into an evolutionary computation algorithm, and two experiments assessing the creation of this workflow. We present our findings as well as possible extensions for this work.

1. INTRODUCTION

1.1 Motivation

Mechanical linkages are mechanisms that are built exclusively from rigid bars whose motion is constrained by hinges, cams, and other types of connections. This project is primarily concerned with a subset of mechanical linkages that are formed by rigid bars connected with revolute joints (also called "pin joints", or "hinge joints") to produce 2D planar motion constrained to one degree of freedom [7].

Passive linkages form the basis of many types of movement in engineered and natural mechanisms, as most physical systems can be reduced to rigid, skeletal bodies constrained by flexible joints. Mechanical linkages are often used to convert one type of motion into another, reduce the amount of force required to perform a motion, or to synchronize the motion of many subsystems.

Linkages are central to many systems that we interact with on a regular basis in unexpected ways. The impact of the combustion engines on locomotion would not have been possible without the linkage that converts the linear motion of the piston into rotational motion to drive wheels and turbines. Mechanical systems of levers and gears formed early predecessors of computers, and linkages with interchangeable cams were used to make the first "programmable" robots.

The range of continuous motions achievable through mechanical linkages is infinite. Kempe's Universality Theorum states that any continuous function can be approximated with arbitrary precision by a planar mechanical linkage [1], leading to the postulation "there is a linkage that signs your name" [8].

However, Kempe's Theorum and subsequent explorations do not give us generalizable rules about how to design simple linkages (consisting of relatively few components) that reproduce desired motion. The current state of rapid prototyping technologies has left us in the following predicament: custom mechanical linkages are relatively easy to *fabricate*, but difficult to *design*. For example, at Disney Research Labs, 3D printing has simplified the process of creating custom mechanical figurines, but complicated software tools and human effort form a bottleneck in the design/manufacturing process [3].

If the task of designing linkages was made simpler and more automated, it would have enormous impact. Passive mechanical linkages can replace powered actuation systems in machine design, reducing control complexity, cost, and power usage. Understanding and designing these systems is useful in many fields outside traditional machine design, whether it is to more accurately design animated characters [12], create prosthetic limbs, or design works of art that turn wind energy into motion [6].

1.2 Approach

The primary variables to be explored in linkage optimization include the *topology*, the specific connectivity of various rigid bars and joints in a given linkage, and *parameters*, the relative lengths of the rigid bars [12]. In this study, we used evolutionary computation to evolve linkage parameters to create linkages that best fit a desired output motion specified by the user.

As the number of bars and joints in a linkage increases, it becomes increasingly difficult for a human designer to accurately predict its motion, and therefore gives us little intuition about how to navigate through parameter space to get closer to a goal. However, curve fitting algorithms and simulated physics environments make the process of checking the output of a given linkage to a desired path relatively simple. This property of being hard to design but easy to test and verify makes linkage design an ideal candidate for evolutionary computing, a process that draws ideas from evolutionary biology to solve complex problems [4].

In evolutionary computing, control of the design process is relinquished to iterative algorithms, which need a relatively quick fitness function for verification, but require less guidance concerning design specifics. Basic examples of evolutionary computing include hill climbing, genetic algorithms, and simulated annealing[5].

In this project, we used evolutionary computing to opti-

mize linkage designs for two different applications: curve-fitting and obstacle course optimization. These are related to both the project tracing the motion of animated characters and creating physical moving structures [12, 6]. Through the formation of this problem as one regarding evolutionary computation, we are able to more efficiently explore the search space and establish parameters that match better with our goals in mind.

1.3 Contributions

The main contributions of this paper include the following:

- Construction of a novel software system that allows users to visualize and interact with the various optimization parameters (Section 3.1).
- Operationalization of the linkage optimization problem as an evolutionary computation problem with regards to representation, variation, mutation, and selection for hill-climbing as well as a genetic algorithm (Section 3.2).
- Experimental validation of the linkage optimization on a curve-fitting problem, including creating a stronger fitness function, exploration of different target curves, and modification of parameters such as population size and mutation rate (Section 4).
- Exploration of the linkage optimization impact on the creation of a walking system composed of a linkage varied by similar parameters (Section 5).

2. RELATED WORK

The related work on this topic spans many fields. Most notably, work at Disney Research Labs has used hill climbing algorithms as well as user-selected fitness to solve for mechanisms that transform rotational motion into animated movement of plastic toys [3]. The Disney researchers describe an iterative process, where the movement of each link is first assumed to be controlled by an actuated joint, then one by one replaced with a suitable passive linkage substitute [2].

The link lengths of Theo Jansen's Strandbeest leg design, which forms the basis of all his kinetic sculptures, was solved by a genetic algorithm [6]. In his program, he defined the fitness of a linkage based on the suitability of the output path as a motion for walking. Some parameters he selected for were flatness and constant speed when the foot is in contact with the ground, high stepping height, and maximum time spent in contact with the ground.

Described in the last section, the mathematical analysis of planar linkages by Kempe and others has demonstrated that a critical basis set of mathematical and logical functions is possible using mechanical linkages alone. This leads to the Universality Theorem of 2D planar linkages - that any continuous function can be approximated by enough linkages chained together [1]. Though the mechanisms produced by following Kemepe's Theorem are incredibly impractical, consisting of hundreds or thousands of links for relatively simple motion, they demonstrate that this design space is rich for computational exploration.

Cabrera et al. describe an algorithm for performing multiobjective optimization on planar linkages for the design of a hand mechanism [2].

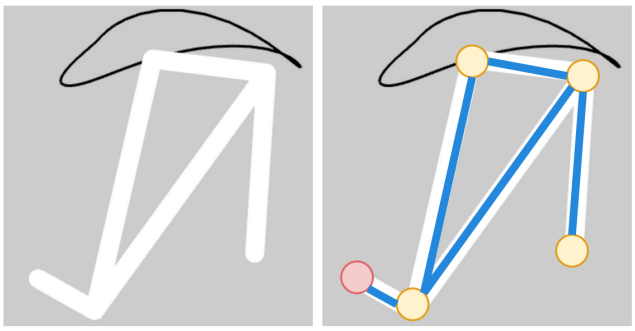


Figure 1: A sample linkage in all white on the left and a diagrammatic break-down showing the major components: links (blue), hinges (yellow), and drive crank (red).

Though the Golem project explores a different design space of linear actuators connected in various configurations, it ties together evolutionary design and rapid prototyping in a way that we hope to explore through our own work [9].

3. METHODS

3.1 Software

The main contribution from this research is the development of a linkage optimization application, *LinkHinge*, which allows a user to perform many types of search over parameter space to find a linkage geometry that most closely reproduces a desired output. This software is written in Javascript, using the libraries/frameworks Backbone.js (UI events), Three.js (WebGL), and Matter.js (2D physics engine).

The latest code can be found on Github, and a live demo of the application is on GitHub Pages.

3.2 Evolutionary Computation Setup

The goal of this project is to reproduce a desired, arbitrary motion path using a 2D planar linkage. We used evolutionary computing algorithms to alter an initial linkage design, specifically its *parameters*, and select for combinations that best fit a given path. The output from our application is a linkage description, including the location and size of each of its components and a description of how to manufacture the design. Some considerations for creating this evolutionary computation algorithm are as follows:

3.2.1 Representation

Though linkages are usually described in terms of rigid bodies constrained by hinges, they can be alternatively thought of as a system of hinges with distance constraints between them. Then, the topology of the linkage describes which hinges are associated with each distance constraint, and the parameters of the linkage are the lengths of the distance constraints. This is the representation used internally within our application. A visual depiction of this representation is shown in Figure 1.

Each linkage owns an array of "hinges" and an array of "links". Some hinges are free to move, and others are fixed. Each link specifies a distance constraint between the two hinges. Links update their position and rotation on the screen based on the position of their associated hinges each

time the physics simulator increments in time.

Each linkage also owns a "drive crank" which causes a hinge to move around another fixed hinge at a constant angular velocity. This rotational motion drives the movement of the rest of the linkage, and since the system is constrained to only one degree of freedom, causes it to trace its output path over and over.

3.2.2 Variation

Variation occurs in two main methods: mutation and crossover. We started with a hill-climbing only algorithm with mutations. Mutation in evolutionary computing is inspired by point mutations in DNA; in our system it causes variation of a single *parameter* of a linkage (the length of a rigid bar).

Then we introduced crossover. Crossover requires populations of more than one, so we switched over to a more traditional GA for this part of the analysis. Crossover occurs between the "links" arrays (described in the previous section) of two parent linkages, causing the children to receive some distance constraints from one parent and some from another.

The crossover used in our system is called "single point crossover", referring to the fact that the links array of each parent is split at a single point and then crossed to form a child array. Single point crossover helps to preserve related sections of a linkage through the evolutionary process. This makes mating less destructive to the optimization process when a design is near a local optimum.

3.2.3 Selection

The fitness functions used to select for fitter individuals is related to the goals for each of the two experiments. Since the goal of Experiment 1 (Section 4) is to trace the target curve using different parameters in the test curve, the fitness function used is based on a curve-fitting algorithm that is robust to changes in phase, rotation, translation, and scale. The fitness function used is elaborated in Section 4.2.

The goal of Experiment 2 (Section 5) is to create a walking system that travels the farthest distance in the shortest amount of time. The fitness function used is based on the distance travelled over a given amount of time. This is further elaborated in Section 5.2.

In the case of both experiments, the selection and mating process is inspired by Daniel Shiffman's *The Evolution of Code* in its setup [10]. Each time a new generation of linkages needs to be calculated, the previous generation are thrown into a mating pool where they are given a probability of mating based on their fitness. Two parents are drawn from the mating pool according to these probabilities to create one child linkage. This process is called "fitness proportionate selection".

3.3 Experimental Validation

As alluded to in the previous section, we use two distinct experimental setups with different goals to test our evolutionary linkage optimization setup. The first experiment (Section 4 looks at curve fitting, or how well our system can evolve linkages that match up with a given input curve. The second experiment (Section 5) looks at how well a mechanical system constructed using many copies of the same linkage performs on a walking task, specifically how far this system can travel over a given amount of time.

For each of the experiments, we tested different parameters of optimization on two different types of evolutionary computation algorithms, hill-climbing and a genetic algorithm. On a high-level, *hill-climbing* is based on random mutations of a single individual linkage system to include only improvements based on the fitness function and the *genetic algorithm* is based on a population of linkage systems that include both mutations and crossover as a means of variation.

Our software application allows for the variation of many parameters, including population size, mutation rate, and maximum length of change. Detailed analysis of these parameters is given in Section 4.3.

In the following sections we present the results of each of these studies using the software constructed and the operationalization of the problem to fit evolutionary computing described above.

4. EXPERIMENT #1: CURVE FITTING

4.1 Study Setup

The first experiment looks at the evolution of linkages to fit a given input curve. The input to the algorithm is a set of points sampled at fixed time steps that describe the motion of interest. At each generation, a fitness function checks the closeness of the generated curve to the input curve in a way that is robust to translation, rotation, scale, and phase of the two different curves through a process of normalization.

We investigate the results of this study based on hill-climbing and the genetic algorithm with the parameters described above.

4.2 Fitness Function

The fitness function of the linkage system in Experiment 1 is based mainly on *curve fitting*, or how well the shape of the motion of the linkage system matches with the desired input motion. This curve fitting measures the difference between the output path of a linkage against the variants of the input curve. We create a fitness calculation that is robust to the following parameters by normalizing the curve in the following manner:

- **Translation.** In order to correct for a curve that is potentially off center, we calculate the midpoint of the two points that are furthest apart on the curve and shift the curve so that the midpoint is on the origin.
- **Rotation.** Correcting for rotation, we calculate the angle between the midpoint and the line drawn between the two farthest points on the curve and rotate the curve so that the angle between the two farthest points is 0 and the line is effectively horizontal.
- **Scale.** The "radius" or distance between the calculated midpoint of the curve and the farthest points is adjusted to be equal to one, shifting all of the other points by the inverse of that distance.

These steps to normalization can be seen in Figure 2. After the curves are normalized, the curves then have to be matched for phase before calculating the fitness. The process by which the curves are matched for phase is loosely-inspired by the RANSAC algorithm and rotates setting each point as the phase anchor. For each of the phase anchors,

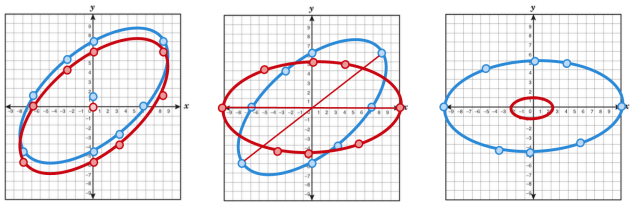


Figure 2: The first three steps to normalization: (1) translation, (2) rotation, and (3) scaling, where blue is the original curve and red is the normalized curve.

```
for each of the points in the output curve,

     find the closest point in the target curve ;
     set this point to be the phase ;
     map the points of the output curve to the points
     on the target curve based on this current phase ;

     for each of the mapped points,
          calculate and store the distance between ;

     store the sum of the distances ;

return the lowest sum of the distances
```

Figure 3: Pseudocode for calculating fitness between an output curve and a target curve in a manner that is sensitive to phase shift.

the fitnesses are calculated, or the distances between each of the points. This process is described in the pseudocode in Figure 3.

4.3 Results

We first start with an assortment of target curves that vary based on complexity, concavity, and size and both hill-climbing and genetic algorithms are run for the optimization of these curves. Then, for genetic algorithms, selecting one of the target curves and using a population of 20 linkages, we alter the mutation rates. Then, taking the best mutation rate based on the above analysis, we alter the population sizes of individual linkages. For hill-climbing we look at a variety of the maximum length of change. The results of these experiments are explored below.

Each one of the figures in the following section graphs the maximum fitness of linkages in a given population at each time step. By graphing the maximum fitness (as opposed to say the average fitness or minimum fitness) we are able to examine the best cases for each one of the iterations, seeing the limits of what the algorithm can output at a given time step. Furthermore, in the end, it is the maximum fitness that indicates whether the goal has been met or not.

4.3.1 Target Curves

The sample target curves can be seen in Figure 4. These target curves are known to be achievable by the given topology because they are generated using the given linkage topology with modified parameters to achieve various extreme cases. The parameters are adjustments to the original location of each of the hinges and therefore the lengths of the links as randomly selected numbers within a small interval near the actual values for the base configuration. These sample target curves are selected specifically due to their

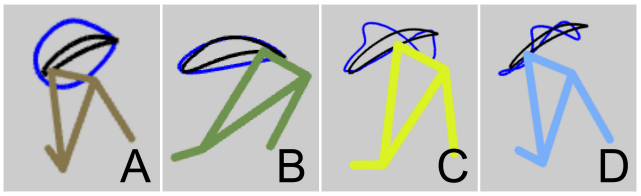


Figure 4: Images of the sample target curves created by the linkages indicated by the bright blue line over the initial output curve outline in black.

differences in size, complexity, and concavity. For example, sample curve A is actually much smaller in size before normalization and sample curve D is the most complex including four loops, or the most concavity as well.

Each of these target curves was tested using both hill-climbing and genetic algorithms for 5,000 runs each, using a mutation rate of 1%. In the case of genetic algorithms, we use a population size of 20 and in the case of hill-climbing, we use a maximum length of change of 15%. The results of these runs are show in Figure 5.

These results show a very interesting trend. Within each of the individual output curves, there is some interesting movement. It seems like for the A input curve, or the "roundest" target curve, since the initial output curve is generally fairly flat and shaped more like input curve B, there is a steady growth from a very low fitness (since all of the points in the flat output curve are pretty far away from the points on the round target curve).

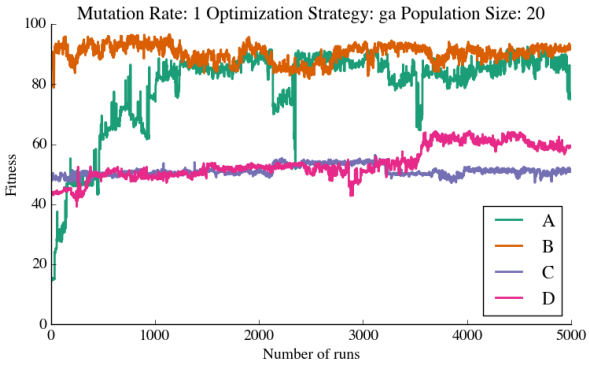
There seems to be a fairly large gap between the two types of curves, A and B which are fairly simple compared to C and D which are much more complicated. As expected, A and B perform fairly well for both optimization strategies, within the 80 and 100 fitness range and C and D sit in the 40 to 60 fitness range. This is interesting because it demonstrates that the fitness function is better optimized for handling smooth, convex curves. This makes sense because those seem to be easier cases to handle and match. However, this is an interesting roadblock and can be explored by altering the parameters of the fitness function to better account for target curves that include more complexity and concavity, specifically looking at the number of times the curve crosses itself or using more drastic changes in the mutation steps.

Comparing between the genetic algorithm and the hill climbing algorithm, we see that while there are some drastic fitness changes in hill climbing, it is the genetic algorithm that presents the stronger levels of improvement. This makes sense because the genetic algorithm approach allows for a much wider range of mutation, encouraging crossover in addition to the smaller point mutations. This then makes sense that there are bigger jumps in genetic algorithm and it is reaffirming to know that most of these jumps will actually be improvements.

4.3.2 Mutation Rates

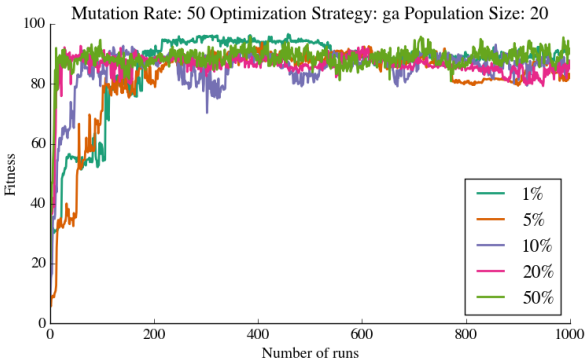
The next parameter that was changed for the genetic algorithm approach was the mutation rates. In this case, we test two different curves, the most simple target curve A and the most complicated target curve D on varying mutation rates, initially of 1%, 5%, and 10%. The results of these runs can be seen in Figure 6.

(a) Hill climbing

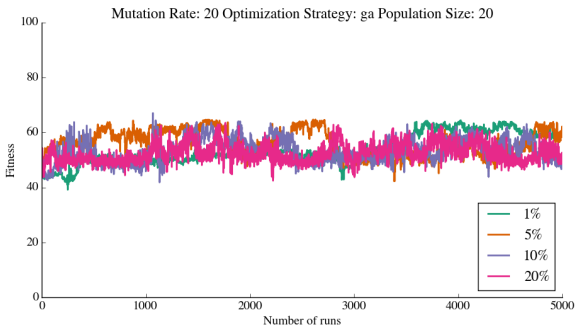


(b) Genetic algorithm

Figure 5: Hill-climbing and genetic algorithm approaches on various sample curves (A, B, C, and D).



(a) Target Curve A



(b) Target Curve D

Figure 6: Performances of various mutation rates (1%, 5%, and 10%) on input curves A and D.

Surprisingly, it does not seem that mutation rate effects the performance of the algorithm on curve fitting. Looking at the results with respect to input curve D, or the most complex curve, as seen in Figure 6b, the different mutation rates seem indiscernible. Not only do they continue to span the same range of fitnesses, between 40 and 60, there does not seem to be a strong trend of any mutation rate doing much better than the others. The two mutation rates that slightly stronger showings with the most time steps that are on the higher levels, or the orange 5% and green 1% curves, both seem to reach these higher states at a relatively similar rate. The main difference is that the green or 1% curve seems to dip a bit lower than the orange 5% curve at some points.

For input curve A, or a simpler curve that the algorithm has traditionally performed at a higher capacity on, the performances past the location of the plateau seem to be somewhat similar. The main differences can be identified in the 0 to 200 runs time step. In this case, it seems like the mutation rate changes the rate a which the plateau is reached. This is seen in the fact that the 1% and 5%, which are proportionally the closest mutation changes, can be seen to plateau together, their curves intertwined in the 0 to 200 time step. However, as the mutation rate doubles to 10%, the plateau is reached much faster. Doubling once again brings the 20% showing to be twice as fast. What is absolutely intriguing is the fact that the 20% and the 50% mutation rates perform at a similar capacity. An explanation for this could be that there is some limit in practice at which the plateau can be reached, which the 20% and 50% mutation rates accelerate towards. Another explanation could be that at relatively high rates, different rates contribute very little in terms of differences in speed.

Therefore, the 5% mutation rate is reasonable for more difficult curves and a higher mutation rate on simpler curves, perhaps a suggested 20% allows the plateau to be reached at a much faster pace.

4.3.3 Population Sizes

Another parameter that can be adjusted for the genetic algorithm approach is the number of individuals in a given population. The previous runs have all used populations of 20 individuals. In these tests, we hold the mutation rate constant at 1% and look at the effect of populations with sizes of 5, 10, 20, 50, and 1000 individuals. The results of these runs can be seen in Figure 7.

These results are very interesting and demonstrate the fact that population size has a large effect on the rate a which the target curves plateau. On target curve A in Figure 7a, the difference is quite obvious. The curves are generally staggered in a way that is positively correlated with population size: the lower population sizes reach their plateaus a lot faster than the higher population sizes. However, one interesting case is the population size 20 versus the population size 50. The population size 20 actually does the second best despite not being the second highest population, crossing the population 50 line and reaching the plateau earlier. And based on the current number of runs, it seems like the different population sizes reach different plateaus.

For the target curve D, seen in Figure 7b, the differences are quite stark but less correlated. For example, the population that seems to do the best and actually breaks out of the otherwise generally restricted fitness of 40-60 is the population size of 50, the second largest population size tested. The largest population size, 100, reaches its plateau the fastest but the plateau is lower than that of the population 50 and is later met by the population 20 and population 5. The population 10 seems to do the worst, actually dipping a few times lower than the original fitness and then ultimately not fluctuating much.

Between the two different curves there does not seem to be a very strong common trend. While it is interesting that there is such a diverse range of results from the varying population sizes, the lack of a coherent trend, except perhaps the slight trend seen between population size and speed of reaching a plateau for input curve A, means that these results are not quite conclusive. The lack of correlation in input curve D could be due to the difficulty of reaching a good solution in general because of the fitness function. Future approaches could test a variety of population sizes in parallel or even randomly generate different population sizes to find a better fit.

The reason why the number of trials is truncated at 500 is because running the population of 100 overnight only resulted in 1048 runs being successfully completed before debilitating my computer and causing it to shut down and delete all of the data. This may change the interpretation of the data as the differences between each of the individual curves is magnified within this relatively short timescale compared to other timescales. This could be traced back to the fact that a lot of the algorithms used are not scalable efficiently. For example, the phase calculations are done on all of the points whereas a smarter sampling of less points in terms of which are used as phase anchors and which distances from the original curve are calculate could probably give a better sense of the fitness.

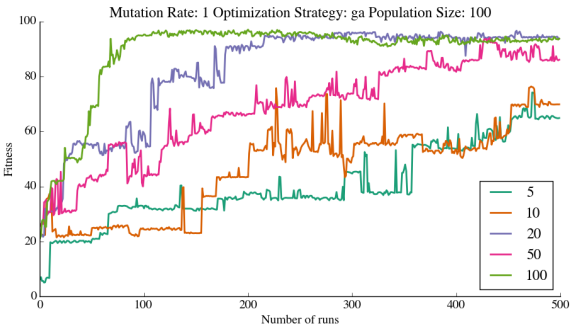
One consideration for future implementation is the efficiency trade-off. Running with such a large population size is computationally intensive, and based on the results and the analysis above, might not be worth it for the most part. A more efficient algorithm could change the trade-off seen and shift the type of approach that is used in the future.

4.3.4 Maximum Length of Change

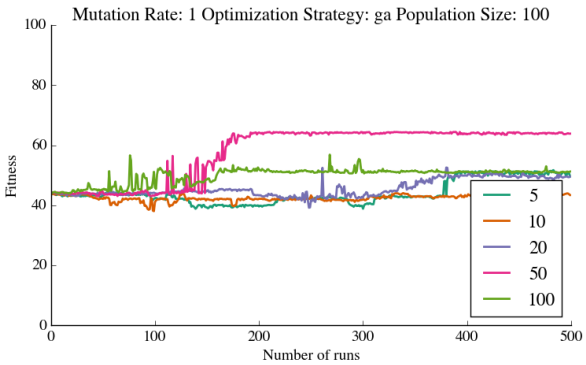
Looking at the parameters that affect the hill-climbing approach, most are centered around the maximum length of change. In these tests, we look at the maximum length of change of 5%, 15%, 25%, and 50%. The results of these runs can be seen in Figure 8.

These are interesting because they demonstrate that the maximum length of change does have some effect on the types of results that emerge from the algorithm. In the case of target curve A, seen in Figure 8a, the various maximum length of changes create a nice spread in the rate at which the plateau is reached. The maximum length of change of 5 seems to perform the best in reaching the plateau at the fastest speed, 25 at the second and 15 and 50 winding together to reach their plateaus at similar speeds. This is interesting because, once again, there does not seem to be a linear correlation between the different maximum lengths and the resulting fitnesses, but it is interesting that the lowest maximum length was able to plateau the fastest. This is potentially due to the nature of the target curve and the fact that the original output curve is already relatively close to the target curve to begin with.

For target curve B, seen in Figure 8b, there is also a wide

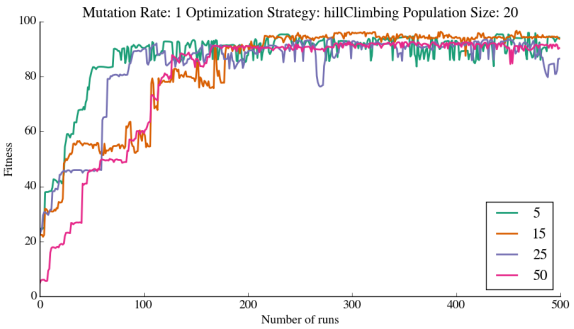


(a) Target Curve A



(b) Target Curve D

Figure 7: Various population sizes (5, 10, 20, 50, 100) on input curves A and D.



(a) Target Curve A

(b) Target Curve D

Figure 8: Various maximum length of change (5%, 15%, 25%, 50%) on input curves A and D.

array of results not necessarily in terms of the rate of the curve in reaching the plateau but in the quality of the results in general. It seems that the maximum length of change of 50 does very well, though the maximum length of 5 catches up later in the time periods. The other ones, populations 15 and 25 do not perform as well as the other maximum lengths. This is interesting because this shows that the "middle" curves may not perform as well as the two "end" curves.

Across the two target curves, it seems that the only trend is that the maximum length of change of 5 seems to perform the best (or in the case of the second, well enough) when compared to the other input curves). One possible generalization is that a lower maximum length of change for simpler curves is better and a larger maximum length of change for more complicated and crazy curves is merited. This makes sense because the crazier curves require more reaches in terms of maximum length because it may require crazier changes in the lengths to generate such different curves. Further examination is necessary, especially with regards to the craziness of the curves and the percentage of maximum changes that are allow (potentially even seeing if 100% maximum length of change would create some differences).

There are some weaknesses to this analysis because hill climbing in and of itself is fairly random, meaning that multiple trials with these different parameters of maximum length of change may alter the output depending on many different factors. Therefore, future work could then look at different iterations with the same parameters to see if there is sufficient variation between the different trials with the same parameters.

Also there, is the weakness of differences in the types of curves. While the fitnesses in target curve A did converge somewhat, this was almost expected, given the performance of target curve A in the other trials. However, it is hard to tell if the fact that target D did not converge is due to the fact that there is a trend in the differences between the maximum length of change or due to the fact that the solution is simply more difficult to achieve. It would be interesting to see how the various trials proceed at a much larger time scale, but due to time and computational constraints, it was not possible to do so this round.

4.4 Discussion

Altering the different input curves, mutation rate, population size, and maximum length of change created an interesting array of results across the different experiments. Some of the interesting insights across all of these different parameters include the large differences that can be induced with just a small shift in parameters. For example, the sample curves very neatly divided into two different types, smooth and loopy curves. While the mutation rate did not have discernible changes across the different parameters on the long term, all of the mutation rates, population sizes, and maximum length of change had some differences in the 500 runs range.

Future iterations of testing would need to focus on a different assortment of target curves of intermediate complexity, more iterations with a given set of parameters until a more acceptable plateau has been reached, and more repeated iterations with a given set of parameters to understand the effect of the element of randomness on the average of the multiple trials.

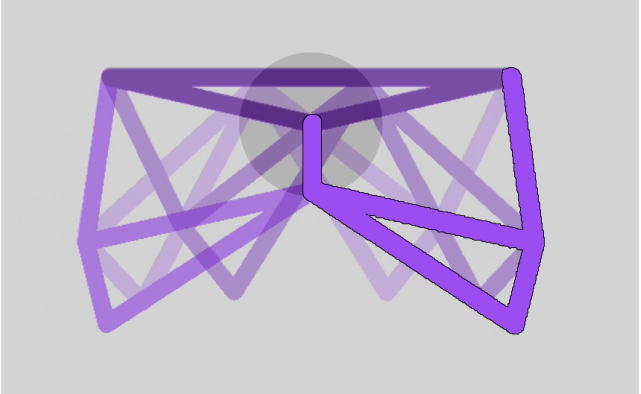


Figure 10: Diagram of a sample walking machine made from an assembly of six leg linkages. One leg is highlighted in bright purple, the other five legs are shown in translucent purple. The legs are connected to a rigid triangular body structure and driven by a central crank at the center of the machine.

Some concerns include the fact that though the graphs were graphing the maximum fitness of the best linkage for each of the experiments, none of the iterations were able to achieve an ultimate "perfect" score of 100 fitness. This is scaled to the difficulty of the curve, seen through the differences between target curves A and B and target curves C and D, but this could also be due to the need to improve the fitness function.

5. EXPERIMENT #2: OBSTACLE COURSE

5.1 Study Setup

In the second experiment, we patterned several copies of one linkage to form the legs of a mechanical walking machine (Figure 10). A pair of legs is constructed by reflecting one linkage of phase θ across the y-axis to create a complimentary leg of phase $\pi - \theta$. Then n pairs of legs are added to the central drive crank, each pair out of phase by $2\pi/n$ radians.

In the experimental setup, a user can define the number of leg pairs used in the construction of the walker; our data is based on walkers made from 3 leg pairs. Fitness was assessed based on the ability of the machine to walk through a simulated physics environment.

This portion of the analysis required an additional dimension in the design space - the rotation of the leg relative to the body. This variable effects both the static stability of the walker and the portion of its output path that comes in contact with the ground. Small changes in leg attachment angle result in dramatic changes in fitness.

5.2 Fitness Function

Instead of optimizing for a user defined path, this experiment optimizes for a task; this has the effect of simplifying the fitness function greatly from the experiment 1. The relative fitness of two walking machines can be assessed easily by racing them across the obstacle course. This portion of the project is more inline with related work by Sims[11] and Jansen[6].

5.3 Results

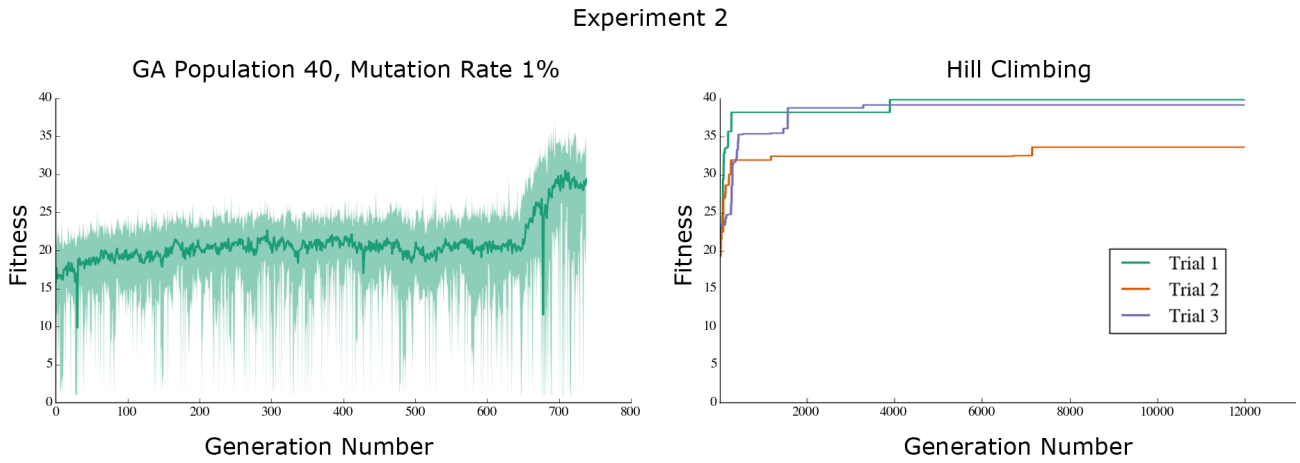


Figure 9: Results from the genetic algorithm run (left) show that the population was able to collectively avoid a local optimum and reach a more globally optimized solution. Results from three walking hill climbing trials (right) show two converged to approximately the same solution and one became stuck at a local optimum.

The results are shown in Figure 9. The units of fitness for this portion of the analysis are speed (distance in pixels traversed per time of walker cycle). The fitness values returned by the hill climbing trials may be directly compared to those returned by the genetic algorithm

5.3.1 Hill-climbing

During this analysis, we ran three trials of hill climbing, allowing the system to run for 12000 generations. Two hill climbing trials (trials 1 and 3) converged on nearly identical linkage designs, ending with a fitness of approximately 40. The second trial converged on a local optimum whose method of locomotion looks starkly different from the other two trials.

The appearance of this local optimum in parameter space is highly correlated to leg attachment angle. If the legs of a walker are attached with a negative angle (causing the body to form a downward facing triangle as in Figure 10), then the system will converge on the solution found by trial 2. Conversely, if the leg attachment angle is positive (forming an upward pointing triangle for the body, as in Figure 11), the system is allowed to converge to the more globally optimized solution found in trials 1 and 3.

5.3.2 Genetic algorithm

The genetic algorithm run was conducted with a population of 40 linkages and mutation rate of 1%; the trial was allowed to run for 740 generations. Figure 9 shows the spread of fitness across the entire generation in light green and the average fitness in dark green.

Throughout the entire trial, the genetic diversity of the population maintains a healthy spread, free of major bottlenecks. The appearance of bottlenecks in genetic diversity indicate that the population size might be too small, the selection process is giving too much of an advantage to the more fit individuals in the population, or that the methods of variation aren't introducing enough new diversity over time.

5.4 Discussion

In general hill climbing runs much more quickly than genetic algorithms because it requires significantly less processing per generation. This explains why we were able to run three long trials of hill climbing in the same time it took to run one genetic algorithm trial. Hill climbing also tends to converge on a solution much more quickly because it has enormous pressure pushing it uphill in its local parameter space. This is clearly shown in the first few hundred generations of all three hill climbing trials compared to the genetic algorithm trial.

One downfall of hill climbing is its inability to escape from local optima once it starts approaching. By contrast, genetic algorithms have less selective pressure driving them towards higher fitness, so they are able to jump away from local optima and reach global solutions. This leads us to the most interesting result from the walking trials:

If allowed to continue indefinitely, it is highly likely that trial 2 of the hill climbing runs would never escape the local optimum it has converged on by generation 12000. By contrast the genetic algorithm lands on the same local optimum for generations 0-650. The maximum fitness of the genetic trial hovers around 25-28, comparable to the 32-33 fitness obtained in hill climbing trial 2. Suddenly, around generations 650-750, the genetic algorithm trial is able to escape this local optimum and move toward the better solution also discovered by trials 1 and 3 of hill climbing. The new maximum fitness lands at 35-36, comparable to the 38-40 fitness achieved in the hill climbing trials.

6. FABRICATION

In any simulation tool that models something about the real world, it's important to validate against experimental results. To increase computational efficiency, we've made some assumptions and approximations, most notably, that these mechanical systems can be projected and sufficiently modeled in a two-dimensional space.

Furthermore, the design optimization described in this proposal is primarily concerned with the degree of freedom

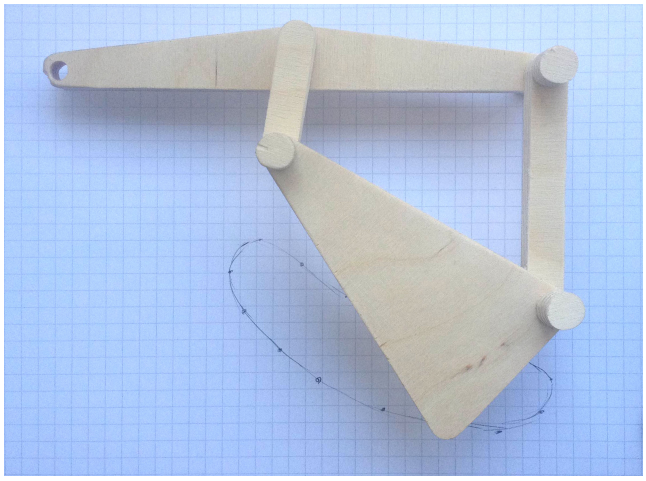


Figure 11: A sample leg fabricated from plywood and aluminum pipe. All wood parts were cut on a CNC router from vector files generated by the javascript app.

constraints of a linkage that result in a given motion. In the real world, other factors such as distribution of mass, torque, 3D structure, and inertia play a role in the viability of a particular linkage design. Some of these factors were accounted for without explicitly simulating them through a physics engine. For example, by imposing a minimum length on all the rigid bodies in a linkage, you ensure that no link in the system requires an exceedingly large force to generate sufficient torque for movement.

To truly test the mechanics of the designs, we used rapid prototyping techniques to realize a select few designs in a non-virtual space. The linkage designs were exported as 2D vector graphics in SVG format from the Javascript application. Then the parts were cut from 1/2" plywood and epoxied to short lengths of aluminum pipe to create working revolution joints. A sample leg is show in Figure 11.

Though development is currently underway, the construction of an entire walking machine was not finished at the time this paper's submission. Supplemental video from the authors will be provided at a later date.

7. CONCLUSION

Our project presents an approach to linkage optimization via evolutionary computing. We utilize the property that the search space of the problem is very large but that the ability to verify the correctness of a solution is fairly simple to establish the representation, variation, and selection involved in the presentation of linkage optimization as an evolutionary computing problem. We assess this workflow using a novel, web-based user interface to evaluate with two different goals: curve-fitting and obstacle course movement. Both problems provide a range of input-driven and free-form problem solving.

Through the curve-fitting evaluations, we find the robustness of the method to various target curves, mutation rates, and population sizes. In the obstacle course evaluation we find that genetic search is able to escape local optima in this design context, but at the cost of processing time.

7.1 Future Work

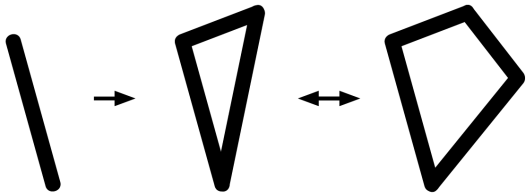


Figure 12: Possible rules for topology mutation.

Ultimately, the goal for this work is to create a design optimization workflow for mechanical linkages. This tool would allow a user to collaborate with computational optimization strategies to generate mechanisms that meet a set of design goals. With only a small basis set of constraint types (explained in detail in Section 7.1.3), it is possible to create mechanisms for a wide variety of mechanical engineering contexts.

7.1.1 Linkage Topology

In this paper we've described search across the parameter space of a given linkage topology to find mechanisms that optimize for curve fitting and task-related fitness. However, one linkage topology generates only a small family of output curves and does not span the range of all possible desired paths. Therefore, to find linkages which are optimized for any arbitrary target path or task, it is necessary to search across linkage topology as well.

Search across linkage topology makes the processes of evolutionary design more complicated. For example, if two linkages have different topology, it is possible that combining them in crossover will be an excessively destructive process. A programmer might decide to separate linkages according to their topology for mating and crossover, analogous to speciation in a biological context.

The mutation of linkage topology requires a set of rules that governs the possible topology changes. One constraint that should be held throughout this process is that the mutation should not alter the combined degrees of freedom of a linkage (combined degrees of freedom always equals 1). Some example mutation rules could be:

- A single link mutates into a rigid three-bar linkage
- A three bar linkage mutates into a four-bar linkage (check that the total combined degrees of freedom of the linkage do not increase)
- A three bar linkage mutates into a four-bar linkage (check that the total combined degrees of freedom of the linkage do not decrease)

These mutation processes are illustrated in Figure 12.

7.1.2 Optimization Strategies

Due to time constraints, we did not get a chance to implement several other optimization strategies of interest into our workflow.

Simulated annealing is a process that allows for larger or more numerous mutations to occur when a solution is

farther from its goal, in order to avoid becoming trapped in local maxima. It is especially useful during hill climbing optimizations, where the system is constrained to only move up in fitness.

Nelder-Mead and Conjugate Gradient are other strategies from combinatorial optimization that allow for efficient search over high dimensional parameter space.

7.1.3 Design Workflow

Eventually, we envision that this design workflow will grow to encompass a wider variety of hinge constraints. So far the tool allows for three types of hinges:

- Fully fixed (0 degrees of freedom)
- Constrained only by the links it's connected to (1 degrees of freedom)
- Driven, currently only driven in a purely rotational motion is possible (1 degree of freedom)

We would like to add the following hinge types:

- Driven on any arbitrary path
- Passively constrained to any arbitrary path (a track follower)

With these five type of hinge constraints, the design tool will be able to model a wide variety of mechanical systems that may interact with the linkage in some way. For example, a driven constraint models the effect of connecting a motor to the linkage; a hinge driven on a line might represent a leadscrew or hydrolic actuator, an arc might be a servo motor, and a circle might be a rotary motor or a turbine, an arbitrary looping path could be a cam or irregular geared system. Passive constraints models track followers that constrain movement of parts of a linkage to a 1 degree of freedom path.

The fitness constraints in the design workflow should also grow to encompass more design objectives. Currently, optimizing for a target path allows a user to control position and velocity of one part of the linkage for its entire motion cycle. Other parameters to be used in the fitness function include:

- A small set of control points of interest that the linkage should pass through (keyframing)
- Orientation of linkage extremities/end effectors as it passes through control points or along entire path
- Velocity along sections of the path
- Max torque required to drive the linkage through its full cycle
- Max forces experienced by all parts of the linkage through its motion cycle
- Space required by the linkage

7.1.4 Experiment #1: Curve Fitting

The results of the curve fitting experiment show the promise of this approach to tackling curve fitting problems in the future. However, in this current implementation, the algorithm is only adept at fitting simpler, convex curves. Of course, the given evaluation is relatively robust but is time constrained. The current target curves that are used for assessment include only very smooth curves (target curves A and B) and fairly squiggly curves (target curves C and D). With more time, a more diverse array of curves would be helpful in seeing exactly the range of motion of the linkages and the flexibility of the algorithm. For example, having curves that are otherwise smooth but have only one bump or one cross as opposed to many squiggles or curves can help determine the range of motion.

The goal of simpler, convex curves is generally sufficient for most curve fitting applications, such as the character movement development curve seen in the Tomaszeweski et. al. 2014 paper [12]. However, since mechanical linkages are so powerful in the range of motion they can achieve, it would be important to test this out in a more extensive fashion and adjust the current workflow to match a wider array of curves. It would be important in future iterations to try to develop a fitness function that would be more robust to increasingly complex curves. This could take into account simple aspects of the complicate curves such as the number of times the curve crosses itself.

It would be interesting to try to understand how changing a given parameter of the linkage is reflected in the curve that is outputted. For example, understanding that adjusting the length of a specific linkage could result in a larger motion in that direction or a given ratio of two linkage lengths would result in a wider range of motion. This would be very different for different linkages and would require much more optimization and understanding of each of the given linkages, which would change the balance between overfitting and creating a procedure that is robust in all different situations. A more generalized solution could be more generalized rules of ratios between the link lengths as opposed to absolute individual link lengths. Or even a fitness function that does not only look at how close each of the points is to the original curve but makes different aspects of the curve discrete, such as height versus width or the number of crosses, allowing the evolutionary computation to proceed as it originally is planned to but have more specific goals to optimize for.

7.1.5 Experiment #2: Obstacle Course

An additional feature of the walker simulation that would be fun to play around with is to introduce different types of terrains to optimize for. For example a terrain with inclines and declines might evolve walkers with more asymmetric gaits. A terrain will many small obstacles may evolve walkers with higher step heights. Some sample terrain types are shown in Figure 13.

8. PROJECT CONTRIBUTIONS

Amanda designed and developed the novel, web-based software infrastructure using the appropriate Javascript libraries, created the infrastructure for and ran the simulations with regards to the obstacle course in Experiment #2, and started on the physical construction of linkages. Joy focused on the curve-fitting experiment, developing the fitness

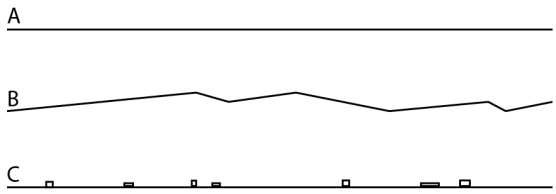


Figure 13: Three possible obstacle courses for walking assessment, flat (A), incline/decline (B), and small obstacles (C).

function through many iterations and ran the evaluations with the different parameters for Experiment #1.

9. REFERENCES

[1] T. Abbott and E. Demaine. Generalizations of kempeâĂŹs universality theorem. *Masters Thesis*, 2008.

[2] J. Cabrera, F. Nadal, J. Munoz, and A. Simon. Multiobjective constrained optimal synthesis of planar mechanisms using a new evolutionary algorithm. *Mechanism and machine theory*, 42(7):791–806, 2007.

[3] S. Coros, B. Thomaszewski, G. Noris, S. Sueda, M. Forberg, R. W. Sumner, W. Matusik, and B. Bickel. Computational design of mechanical characters. *ACM Transactions on Graphics (TOG)*, 32(4):83, 2013.

[4] L. N. De Castro. *Fundamentals of natural computing: basic concepts, algorithms, and applications*. CRC Press, 2006.

[5] A. E. Eiben and J. E. Smith. *Introduction to evolutionary computing*. Springer Science & Business Media, 2003.

[6] T. Jansen. Strandbeests. *Architectural Design*, 78(4):22–27, 2008.

[7] D. Jordan and M. Steiner. Configuration spaces of mechanical linkages. *Discrete & Computational Geometry*, 22(2):297–315, 1999.

[8] H. C. King. Planar linkages and algebraic sets. *Turkish J. Math*, 23(1):33–56, 1999.

[9] H. Lipson and J. Pollack. Automatic design and fabrication of robotic lifeforms. *Nature*, 406:974–978, 2000.

[10] D. Shiffman, S. Fry, and Z. Marsh. *The nature of code*. D. Shiffman, 2012.

[11] K. Sims. Evolving virtual creatures. *SIGGRAPH*, 1:15–22, 1994.

[12] B. Thomaszewski, S. Coros, D. Gauge, V. Megaro, E. Grinspun, and M. Gross. Computational design of linkage-based characters. *ACM Transactions on Graphics (TOG)*, 33(4):64, 2014.