# Project 1: Cannon Evolution

Gavin Haynes & Samuel Beal

https://github.com/gavinsh32/cannonevolution

## Abstract

This paper presents a project where cannons are evolved using fitness algorithms based on an array of two genomes, tilt and power, to hit a target. We generated a population of random cannons, selected cannons by a range of distance to the target and reproduced those cannons with their tilt and power randomly mutated by a small percentage. We then culled off a small portion of the initial population and joined the rest with the children. Through this we were able to evolve a population of cannons to hit a target more often over each generation. We found that lower population sizes reached higher fitness levels in slightly less generations, and higher reproduction rates reached higher fitness levels in significantly less generations than lower reproduction rates.

## Description

We are evolving cannons to hit a target. The genome is comprised of two genes: tilt and power, which will affect the tilt and power of the cannon respectively. Each gene is made up of As and Cs, with As increasing the value of that gene and Cs doing nothing. Our environment is made up of cannons at a specific point trying to hit a target sufficiently far away. This is done through evolution of tilt and power, such as requiring low tilt and high power or high tilt and medium power. Then, successful cannons are cloned and mutated to create a variety of fit entities.

## Reproduction

After fit individuals are selected, they asexually reproduce to form a population of children. The initial population is culled, children are mutated, and the resulting sets are joined to form the next generation. Fit individuals are selected by increasing the total population size by some percentage, currently at rates between 5-10% to mimic biological processes, and copying the clone children to a new population. These children are then mutated as described in the following section and then joined to the total population.

## Mutation

Both genes are mutable, which accepts as a parameter two percentages: for the maximum variability within that gene, and for a percentage of the population. In each, individuals and gene positions are selected using Random. For everyone, the mutation function first determines the discrete number of positions and individuals to mutate, and to introduce more variability.

For everyone, the function goes through the desired gene, and for the maximum calculated frequency, flips an A to a C or vice versa.

For a population, the function determines from a percentage how many individuals mutate and applies to each from there.

## Algorithm Descriptions

- Fitness or our success rate is measured through how close a cannon comes to a target. We currently have a threshold of being within 10% of the targets boundaries which are default set to coordinates (50,50) with a width of 10, where the environment is 100 by 100.

- Cannon entities are generated with two random genes: tilt and power. Each is comprised of a list of 90 characters, A or C, with A's increasing the magnitude of that gene. For each we are adopting a value of 1.0, so for every A, the tilt or power is increased by 1.0. For both, we are using the Python random generator.

- Mutation is applied through selecting between 0 and n characters of the genome of each cannon. Both will be mutated separately, as too high of a rate resulted in extinction.

- We select cannons that come within that 10% threshold of the target's boundaries, and then we reproduce those cannons by a modifiable reproduction rate with a default of 10%.

- We take the children and grab a random percentage of the children based on the passed in mutation rate then individually mutate their tilt and power.

- We mutate them by randomly selecting an index within the population and assigning a random A or C gene. This is done to the percentage of the cannons in the population passed in as explained above.

- We have a square range of (x,y) coordinates as a target, which if the cannonball enters the targets range or within 10% of it by default, it is successful. We check these bounds every time tick as the ball moves across time.

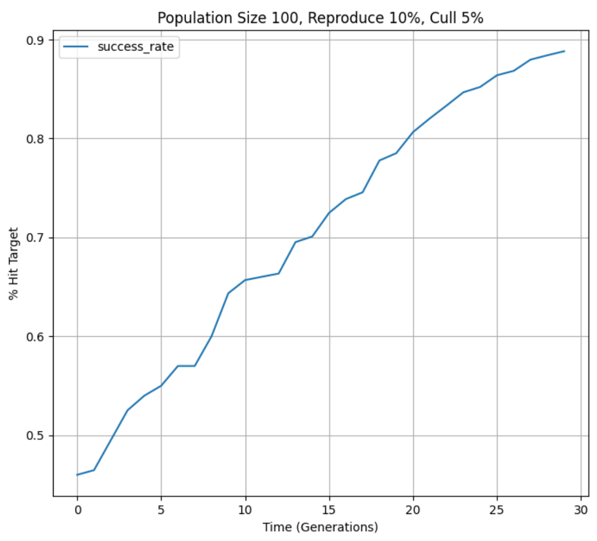
## Experiment 1: Base

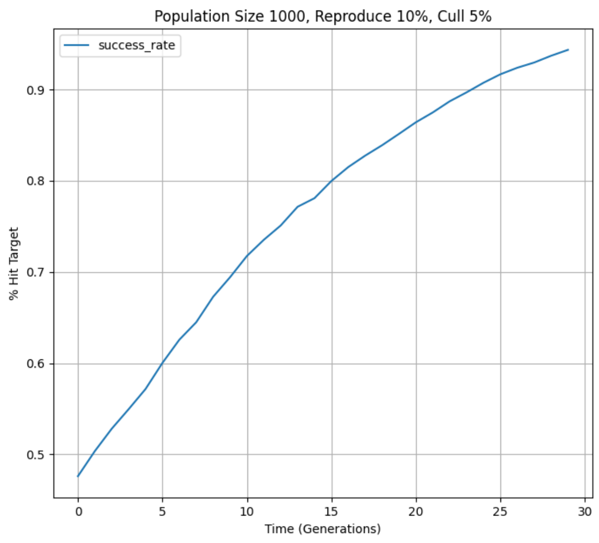
### Setup

This experiment will be a placeholder with some base parameters. This experiment adopts the following:

- Population size = 100, 1000
- Simulator dimensions = 100 x 100m
- Target = 10 x 10m at 50m, 50m
- Tilt mutation: 1 genome
- Mutation Rate: 3% tilt, 2% power
- Reproduction Rate: 10% of the population
- Cull Rate: 5% of population
- % hit target: success rate over time across 30 generations in one run

### Results





### Discussion

We see that over the number of generations our cannons get more and more accurate towards hitting the target. We also notice that with a smaller population size we have more inconsistencies and movement in the graph, and the larger population got more successful faster.

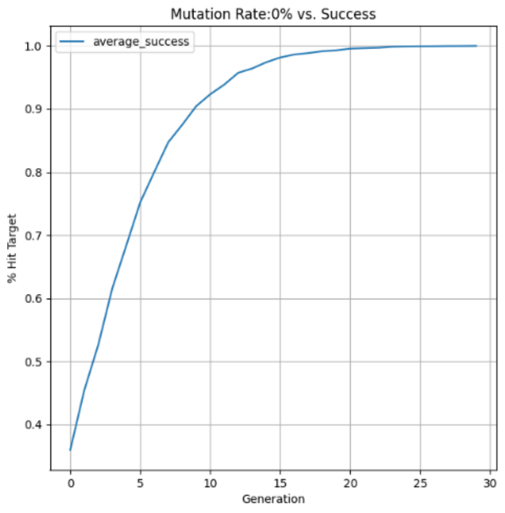
## Experiment 2: Mutation Rate vs. Success

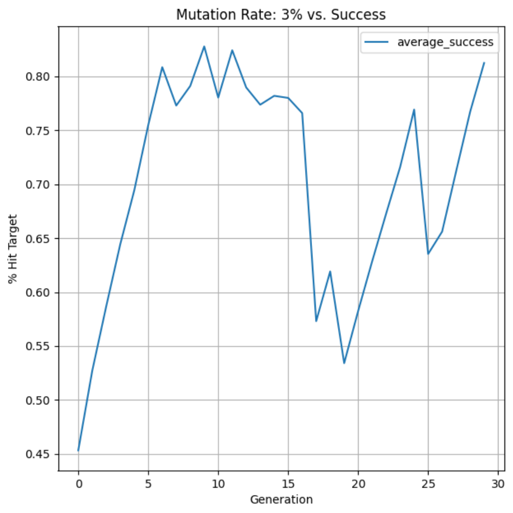
### Setup

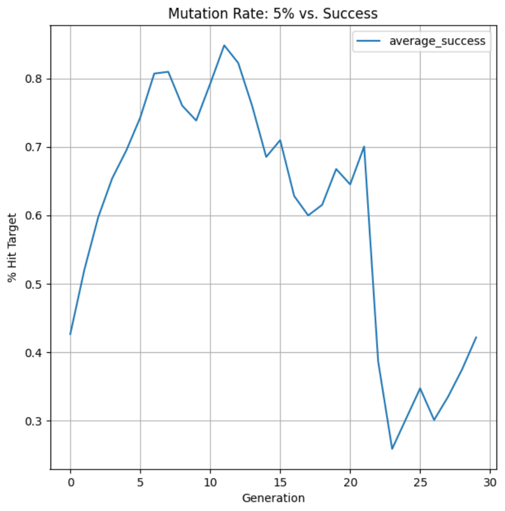
This experiment will test the effect of mutation on the rate of success. This experiment adopts the following parameters:

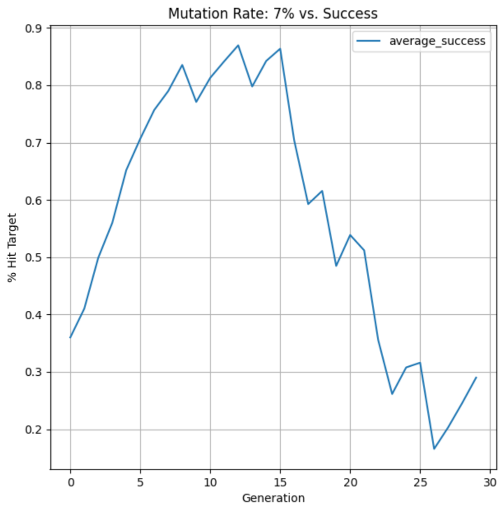
- Population size = 100
- Simulator dimensions = 100 x 100m
- Target = 10 x 10m at 50m, 50m
- Tilt mutation: 1, 2, 3, 4, 5%
- Mutation Rate: .033% of alleles
- Reproduction Rate: 10% of the population
- Cull Rate: 5% of population
- % hit target: percent average at each generation across 3 runs which hit the target

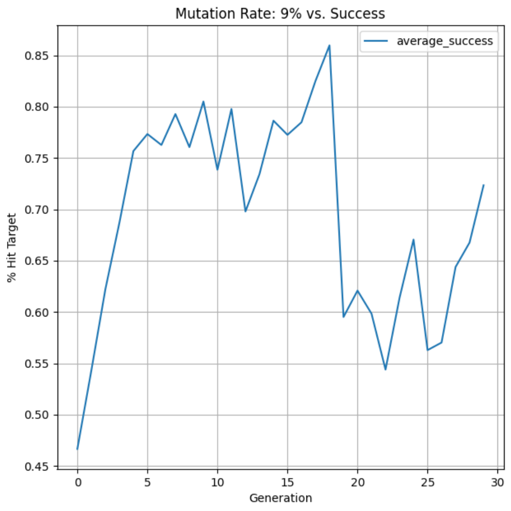
### Results











## Discussion

With low mutation rates, populations on average quickly progress, leveling off at near 100% accuracy towards later generations. For mutation rates 3-9%, while populations do become fit, they are more unstable, either resulting in extinction, or cycling about 50% accuracy.

This may be due to the significance of mutation; each allele is responsible for a change in 1 for the cannons, and for the experimental conditions used, could have a drastic effect on its ability to hit the target. This effect becomes more pronounced in future generations and results in extinction.

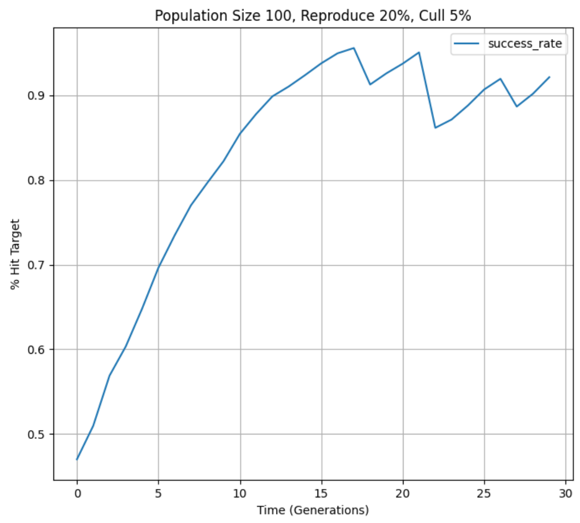
## Experiment 3: Size vs. Success

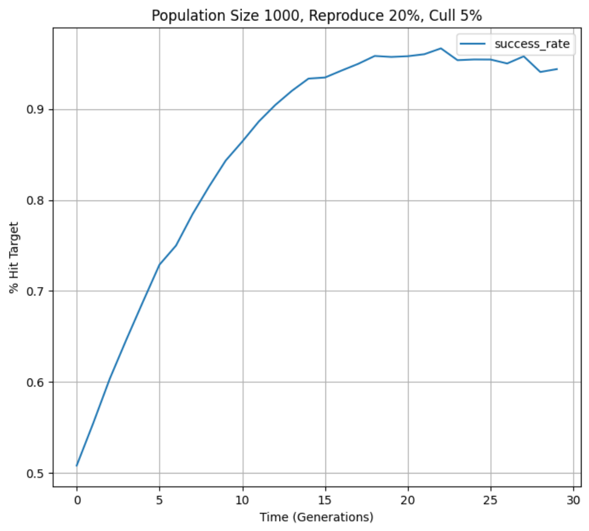
### Setup

This experiment will take our base experiment and test 20% reproduction rate. This experiment adopts the following:

- Population size = 100, 1000
- Simulator dimensions = 100 x 100m
- Target = 10 x 10m at 50m, 50m
- Tilt mutation: 1 genome
- Mutation Rate: 3% tilt, 2% power
- Reproduction Rate: 20% of the population
- Cull Rate: 5% of population
- % hit target: success rate over time across 30 generations in one run

### Results





### Discussion

It appears that reproduction rate drastically increasing the speed at which we have a higher % of hit targets. If we look at our base experiment, we are almost 10 generations faster to reach 0.9. We also see that with the smaller population we have a lot more jumps after we reach 0.9 success rate, showcasing the random gene selection at the end.