Evolving Creatures with the Principles of “Survival of the Fittest” in 

Max Brenner, Thomas Stanton, Ketan Kapse, Dev Shah

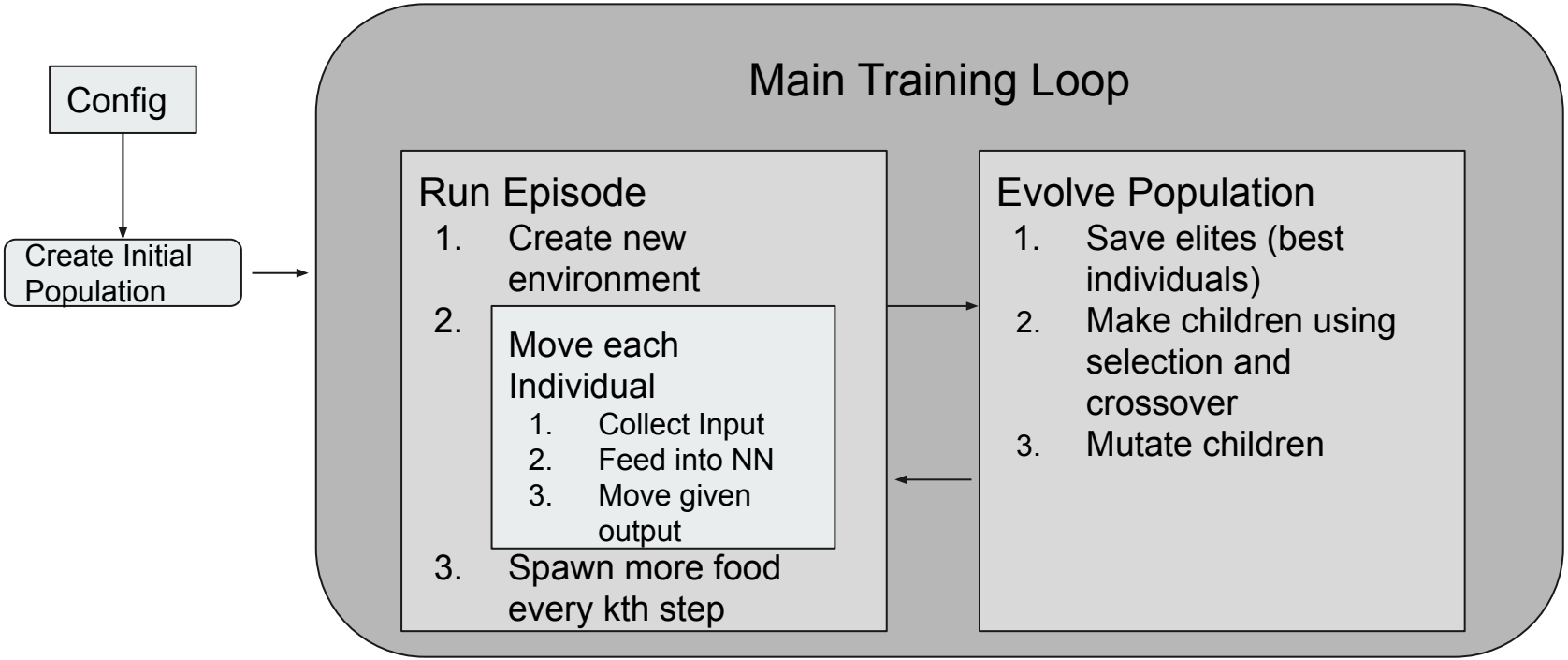
Contents

Project Description and Code Flow

High-Level Project Description

- Program evolves virtual creatures (individuals) to effectively move and gather food in an environment (survival of the fittest)
- A genetic algorithm is used to evolve the chromosomes of the individuals
- The chromosome is the weight values of a feed-forward neural network used to take in an individual's observations about the environment and translate this to a useful movement action

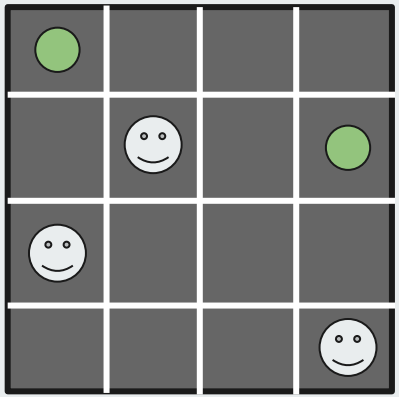
Code Architecture



Concurrent Episode Loop - Detail

Create Environment

- Spawn food, place individuals
- Example: 4 x 4, 3 ind, 2 food



Concurrently Move Individuals

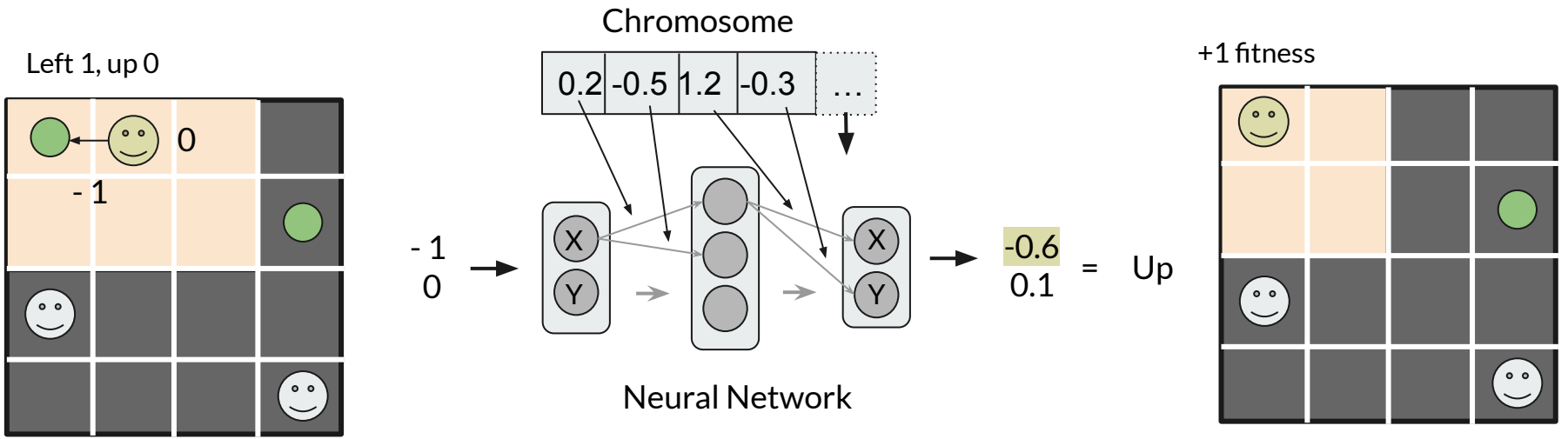
- Movement loop

```
Threads.@threads for individual in population
    move_individual(individual, environment)
end
```

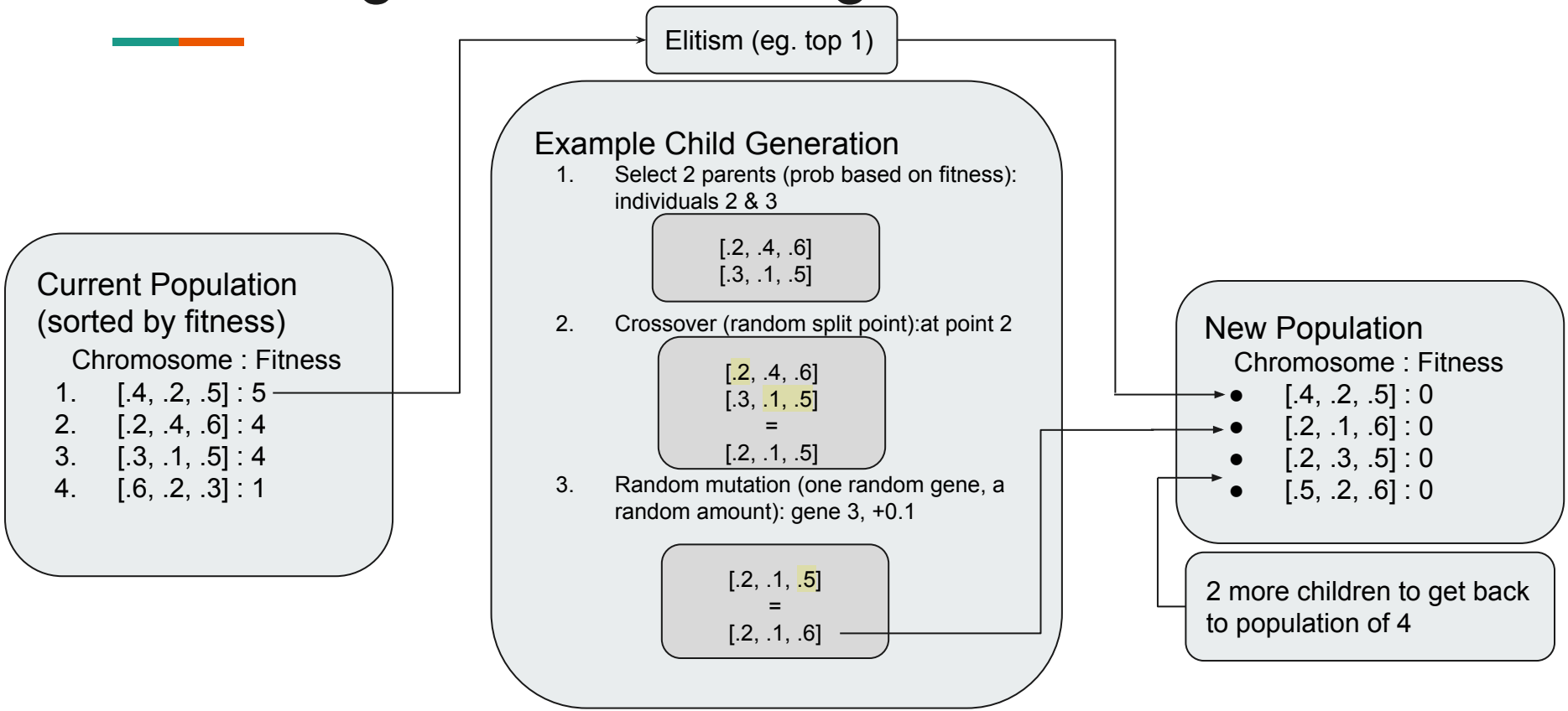
- Respawn eaten food every so often

```
if time_to_respawn
    respawn_food(environment, amt_eaten)
end
```

Individual Movement using a Neural Network - Detail



Genetic Algorithm for Evolving Individuals - Detail

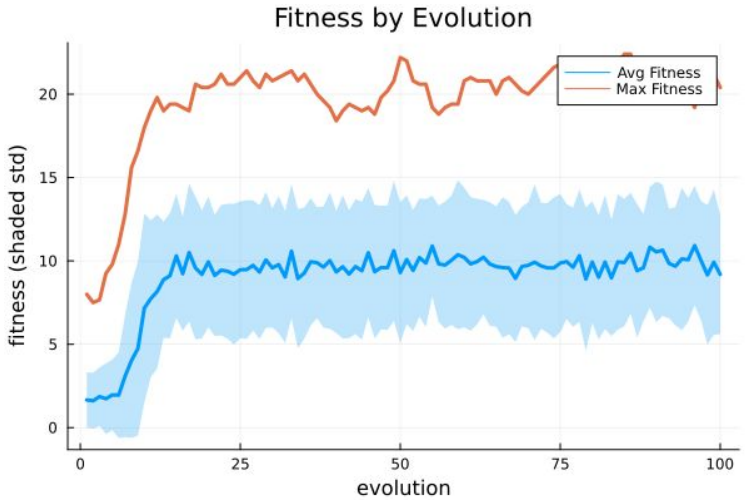


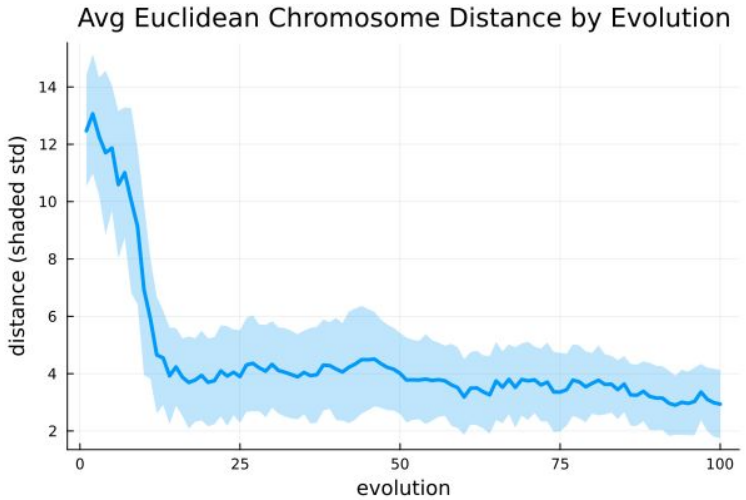
Demo

Experiment with Good Performance

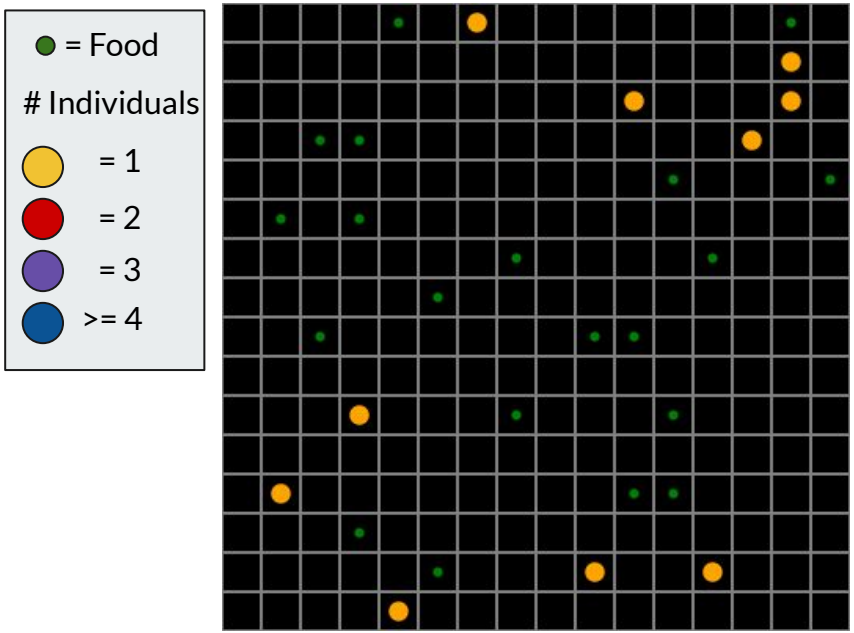
Specifications and Results

- Experiment info: Took ~2 minutes, 100 evolutions, best avg fitness at evolution 96 (10.92)





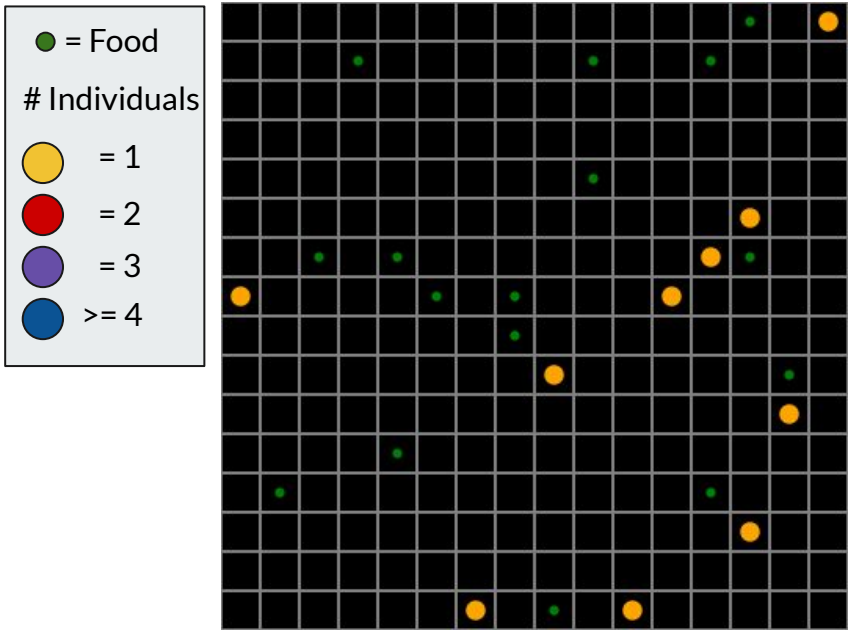
Episode 5 Performance Evaluation



- Evaluated after 5th evolution
- mean fitness: 2, max fitness: 6
- Reduced size of environment: top 10 individuals

- Observations: Not great, some stuck at walls, oscillate between food

Episode 100 Performance Evaluation



- Evaluated after 100th evolution
- mean fitness: 18.7, max fitness: 32
- Reduced size of environment: top 10 individuals

- Observations: Do much better, but some clump together, or still get stuck occasionally

Future Work

Future Work

- Increase performance of best individuals
  - More input types (neighboring individuals, etc.)
  - Longer training
  - Hyperparameter tuning (parent selection algorithm, mutation frequency, etc.)
- Increase complexity of environment
  - More objects (traps, internal walls)
  - Cone of vision instead of circle of vision
- Further speedups
  - More effective collision detection with food
  - Use all multithreading possible if results in faster runtime

Questions?