It’s all MOOT

An MPI adventure.

Brian F. Haag
CSE 633
Dr. Miller

The Problem

- Calculating basic genetic algorithms on a cluster.

- Variable dataset, size of a population varies over time.

- Ultimately though, the data is very small, but needs to be updated very frequently.

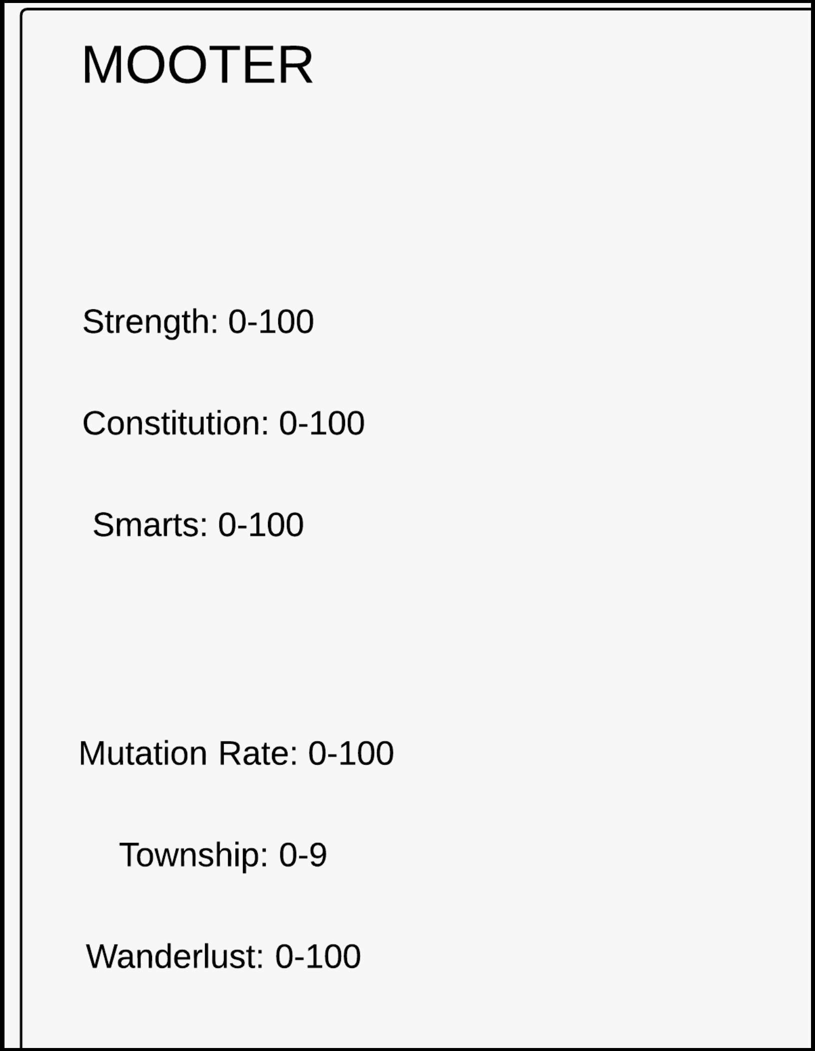
Challenges

- Learning C, I’m most comfortable with LISP, but that didn’t pan out, so I switched early.

- Keeping within scope, runtimes for GAs can spiral out of control quickly.

- Improving performance from initial versions.

What is a Mooter?



MOOTER

Strength: 0-100

Constitution: 0-100

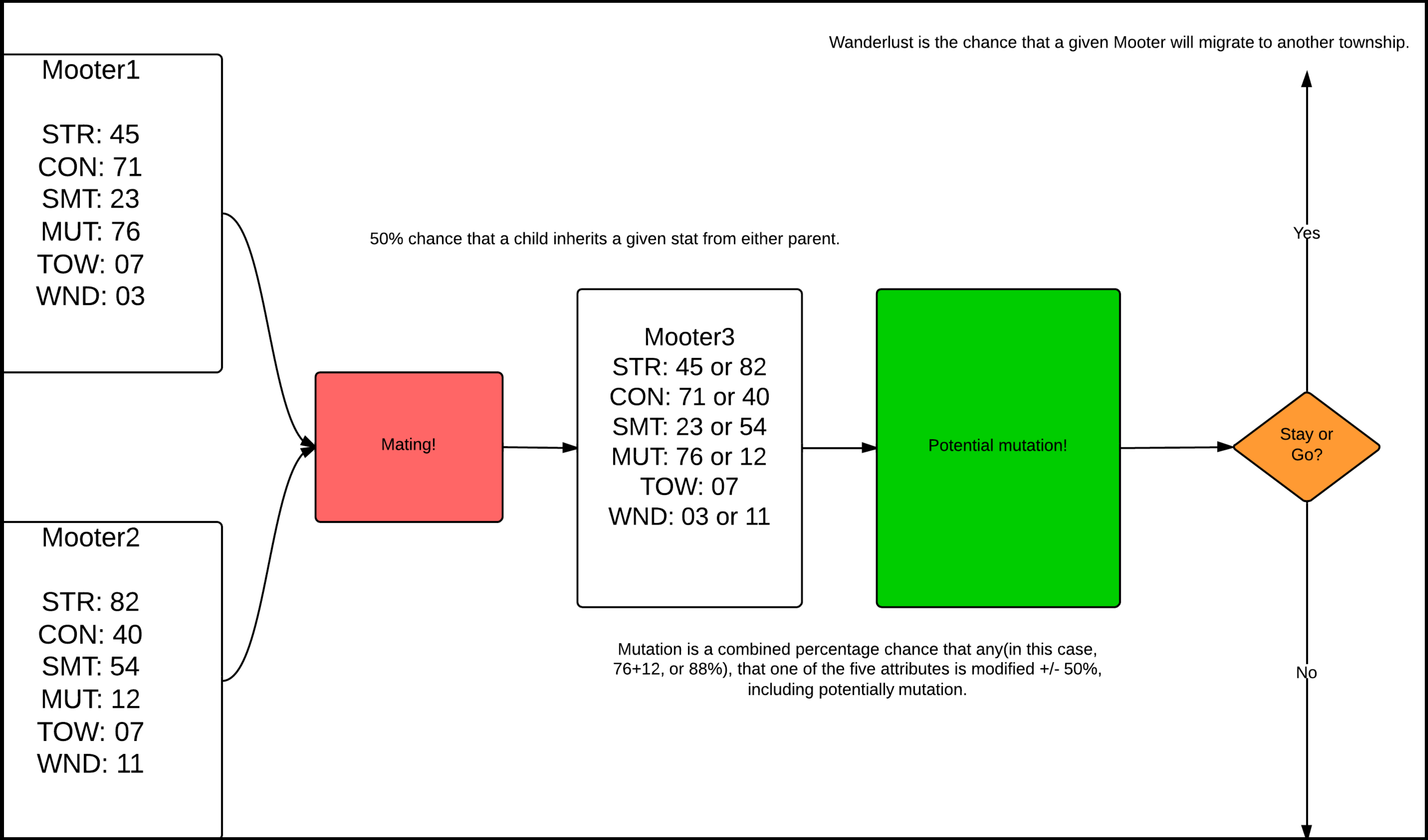
Smarts: 0-100

Mutation Rate: 0-100

Township: 0-9

Wanderlust: 0-100

My parent’s didn’t explain it to me like this.



Wanderlust is the chance that a given Mooter will migrate to another township.

Mooter1

STR: 45
CON: 71
SMT: 23
MUT: 76
TOW: 07
WND: 03

50% chance that a child inherits a given stat from either parent.

Mooter2

STR: 82
CON: 40
SMT: 54
MUT: 12
TOW: 07
WND: 11

Mating!

Mooter3
STR: 45 or 82
CON: 71 or 40
SMT: 23 or 54
MUT: 76 or 12
TOW: 07
WND: 03 or 11

Potential mutation!

Stay or Go?

Yes

No

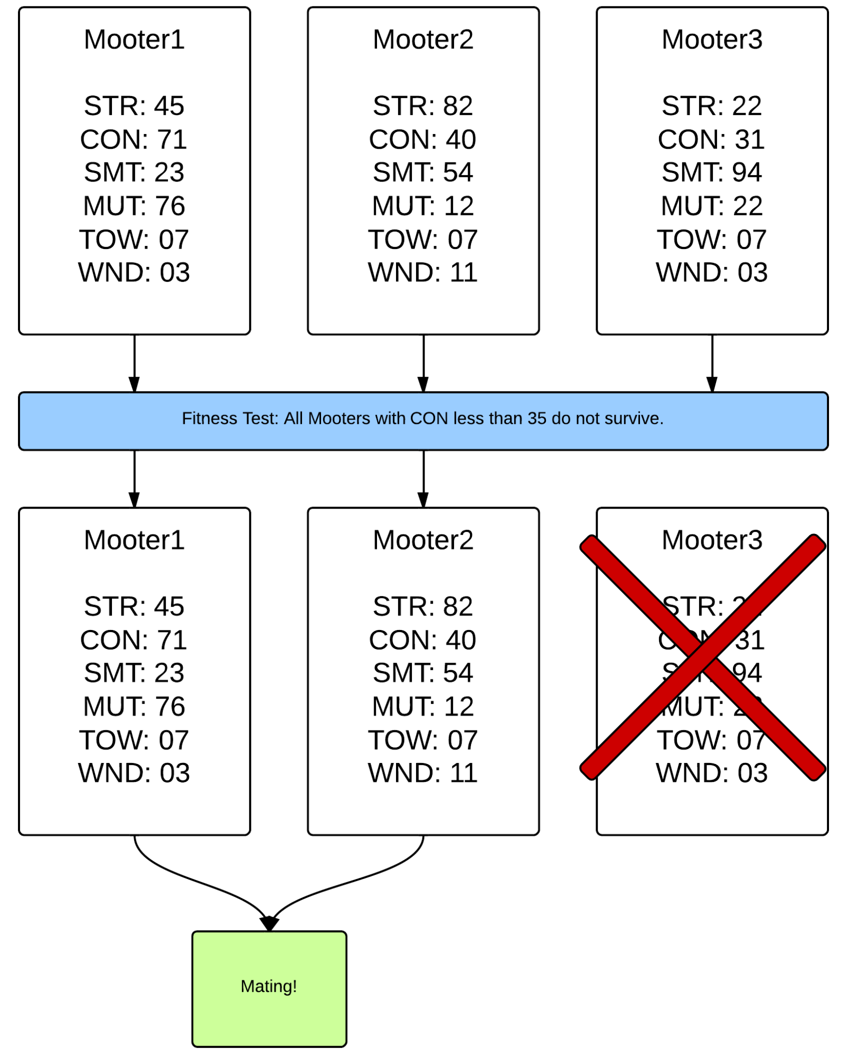
Mutation is a combined percentage chance that any(in this case, 76+12, or 88%), that one of the five attributes is modified +/- 50%, including potentially mutation.

Fitness and Goals

- The fitness goal is to have a population where 80%+ of a population greater than the initial population survive fitness tests for 5 consecutive fitness tests.
- A fitness test is a test against the stats of a given Mooter, if the Mooter ‘s stats are not up to snuff, then they do not survive and their genes do not continue forward.

- Every x generations, Mooters are not limited to their township in terms of potential mating partners(Great Moots).

Example



Mooter1

STR: 45
CON: 71
SMT: 23
MUT: 76
TOW: 07
WND: 03

Mooter2

STR: 82
CON: 40
SMT: 54
MUT: 12
TOW: 07
WND: 11

Mooter3

STR: 22
CON: 31
SMT: 94
MUT: 22
TOW: 07
WND: 03

Fitness Test: All Mooters with CON less than 35 do not survive.

Mooter1

STR: 45
CON: 71
SMT: 23
MUT: 76
TOW: 07
WND: 03

Mooter2

STR: 82
CON: 40
SMT: 54
MUT: 12
TOW: 07
WND: 11

~~Mooter3~~

~~STR: 22~~
~~CON: 31~~
~~SMT: 94~~
~~MUT: 22~~
~~TOW: 07~~
~~WND: 03~~

Mating!

Technique

- The population is scattered through the cluster, with each cluster taking a portion relative to the size of the population
- Wanderlust changes are broadcast out
- Initial population is static to making testing easier.

Observations

- Increasing starting wanderlust has a dramatic (negative)impact on performance.
- Modeling parallel GA using “realistic” scenarios is actually pretty damn hard, there are a lot of potential sequential steps once you start dividing the population(well giving them some mobility).
- Cont

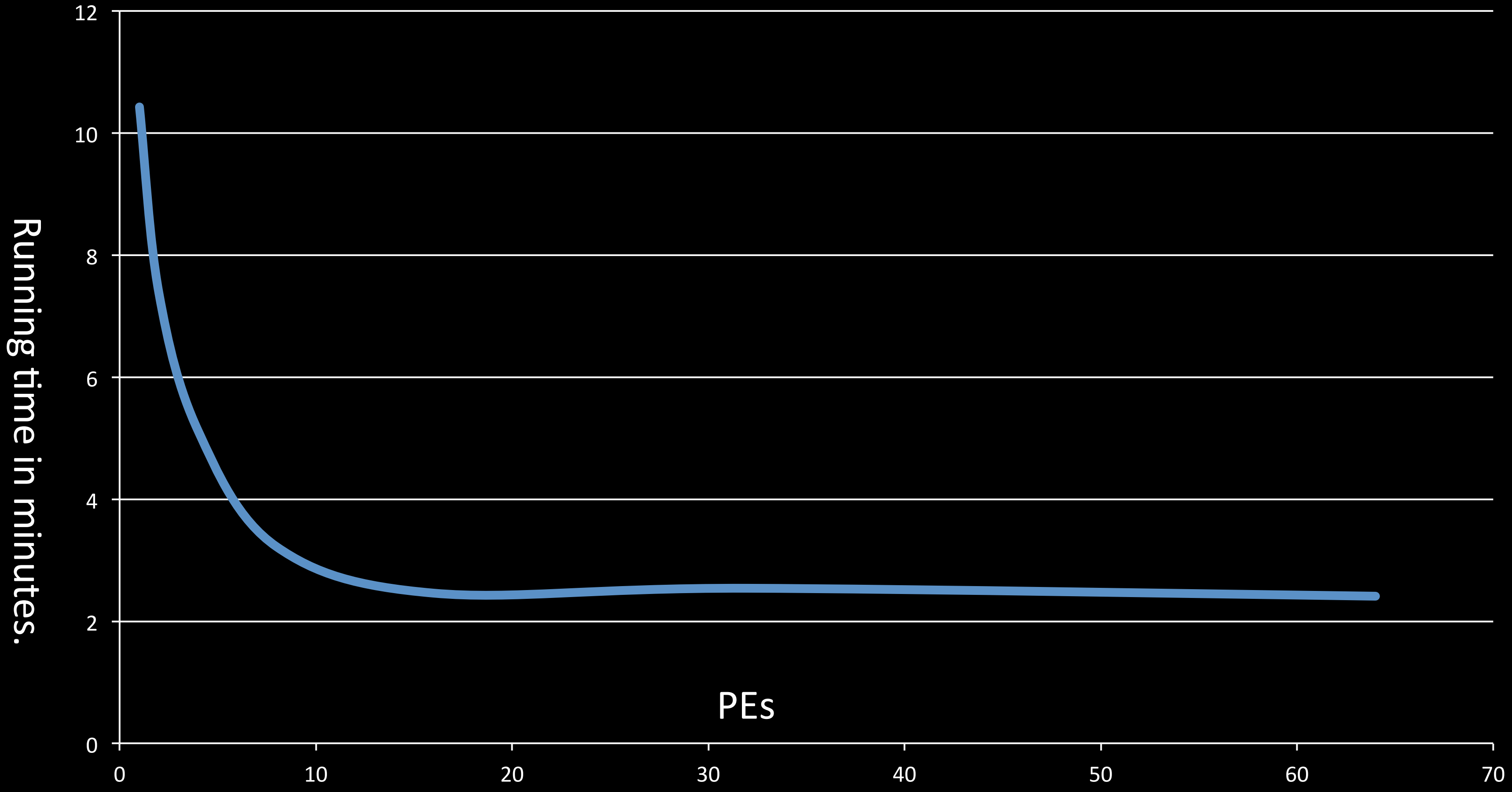
Observations Cont

- Increasing the initial fitness thresholds can result in dramatically speeding up the goal.
- Increasing the rate of Great Moots will increase performance
- Implementing a parallel sorting mechanism would likely improve ability to scale program

1 core per nodes

Initial Testing

Moot Rate: 100 generations
Population Size: 100k



Running time average of 5 tests.

PE	
10.42	1
7.38	2
5.12	4
3.22	8
2.46	16
2.54	32
2.41	64