CSCI4830: Concurrent Programming
Project Checkpoint: Concurrent Genetic Algorithm
Zachary Doyle [zado5040]

Changes to Project Structure

The model problem for the genetic algorithm to solve is has been greatly simplified. Rather than graph traversal (which presented the unpleasant problem of how to "merge" answers between fit competitors for the next generation), Evolvers attempt to guess a number between 0 and some N; all Evolvers in a given Ecosystem are permitted some distance to "walk" from their initial guess (usually N/(10^a) for some a). If their "walked" answer is within a certain threshold of the actual answer (currently, this threshold is static, but this will change, see below), that Evolver survives to reproduce, and produces C children with a random other Evolver, whose intial guesses are the average of the two parents, plus some random genetic transmission error.

Current Implementation

The current, sequential implementation of the project works as follows:

1. An ecosystem is created.
2. The ecosystem is populated with Evolvers in a LinkedList, starting with random guesses.
3. Generation Loop:
    1. Each Evolver, starting from its initial guess, "walks" a limited amount toward the correct answer; while it is not aware of its precise value, it can tell whether

its guess is too low or too high.

2. Evolvers report their final guesses to their Ecosystem. The Ecosystem culls all Evolvers whose guesses are not sufficiently close.
3. Surviving Evolvers are recombined “sexually” (their single genetic 'word', their initial guess, is averaged between two parents and then fudged with some error) to produce a new LinkedList of children.
4. The current generation list is replaced by the child list.

4. The loop terminates when all Evolvers are dead (failure) or the correct answer is guessed (success). In the case of failure, may jump back to Step 2 if desired.

One change that needs to be implemented but is not is making the survival threshold dynamic; currently, it mirrors the walk distance for the ecoystem, but should ideally allow the fittest M Evolvers to survive (i.e., the threshold should be a mean-like value). In general, the Ecosystem will need to be as self-regulating as possible, as its name might imply.

Sequential Implementation Details

The primary shared resource that must be carefully handled with concurrency is the Ecosystem, and its LinkedList of Evolvers. Each Evolver will be given its own thread with which to do its “walking”; the trick will be to only work with the guess data (and therefore start removing/replacing Evolvers within the Ecosystem, once all Evolver

threads have been joined.

Testing Methods

The primary ways I'll be assessing performance are deadlock/starvation-free-ness, as well as actual completion time. To be successful, the parallelized algorithm must be more efficient (on a sufficiently multi-core'd machine) than the sequential one, and not permit deadlocking or thread starvation.

Execution Screenshot