Observations

# 02-02-2017 Notes

## Default Settings Optimization

It is my observation the following settings prove highly effective:

* -p 100
* -o 10
* -r .05

Heretofore, the above settings shall be the defaults.

## Sources of Improvement

One important change, which seems to have improved the outcome of these settings, is the use of the *convergeValue* Decider, which depends upon the convergence of a common value across the population, rather than a common genome.

Additionally, the “radioactivity” setting now operates on a bit-to-bit basis, rather than merely creating a single mutation in the genome per generation. This may be the greatest source of improvement. Previously, if two different, competitive genomes both possessed the same value, and if each differed by more than a single mutation, it was unlikely that either could mutate sufficiently to converge. Similarly, if one configuration rendered a good, but suboptimal result, if the optimal result required more than one mutation to reach, and if all intermediate mutations rendered an even worse result, the genome would never “evolve” to the preferable state. By allowing multiple mutations per generation, this issue is somewhat overcome.

# 02-03-2017 Notes

The options:

* -p 16
* -c 7
* -r .05

Prove highly effective.

## Parent / Child Defaults

Because an array is being used to manage references to different genomes, I suspect that sizing the array to be a power of two might be better for the JVM. I have no idea if this is true, but it seems reasonable. Therefore, I’ve refined the defaults population size and children.

Note that the array is sized as follows:

In other words, as *-c*, or *--children-num*, should be set to 7, 15, or any number such that for some arbitrary value k. Then, *-p­*, or *--population-size*, should be set to 8, 16, 32, or any number such that for some arbitrary value k. This ensures that the final array size is a power of two.

I’ve chosen 16 and 7 because , leading to an array of length 128. Should I have chosen 8 instead of 16, the array would have been on length 64, because .

## Radioactivity

Small populations benefit from a high degree of radioactivity. Sets of 24 items find optimal answers quickly with a radioactivity of 25% or more, and the diversity created by the high radioactivity seems to create more consistently accurate results. High diversity, in conjunction with a large population size, helps ensure that the algorithm does not converge on a suboptimal answer.

# 02-07-2017 Notes

## General Updates

The convergence functionality has been implemented. In retrospect, it is a bit gross that mutation is handled in a static function when an object method is much more sensible. This would have avoided some needless clutter in the Phase Library, and is more philosophically palatable. Objects store state, and methods exist to modify that state. The arbitrary mixing of paradigms is less than preferable, and may be changed in the future.

The Phase Library has begun to outlive the utility of its name. Future versions may revise this name to something more semantically valuable.

## Convergence System Notes

The convergence system (implemented in the “most fit” function in the Phase Library) seems to radically improve the consistency of the function, and works best when given a high number “convergence number” to meet before accepting a solution. I’ve found that 7 is an effective value for most cases. As the function encounters various solutions, no accepted solution will ever be worse than the best result yet found. Suboptimal results are ignored when tallying the number of results consistent with the best solution found thus far.

For file F2.