Sam Trenter
Project 4

For each “gene” in the compact GA, I adjusted it plus or minus 1/populationSize per generation. Larger populations make the CGA run much slower but make it much less prone to converge to a zero.

Try it on onemax and compare to sGA

CGA:
For string size 20 it works very fast, although it will sometimes not converge to all the string (IE one will go to zero instead of one)
For 30, with a large enough population, all converge but with a population size of 100, it seems to struggle. I think there should probably be a scaling difficulty depending on how close you are to one or zero.
40, again works with a large enough population. We really only fail to converge on the correct answer when the population size is too big.

SGA:
For every single one (20,30,40) we converged with every single group. Which makes sense because of how easy oneMax is. We also always converged in fewer generations, but in a longer amount of time with the SGA.

Try it on trap-4 and compare it to sGA

CGA:
For all sizes, none would converge to the stringsize. Sometimes one “genes” will converge to one but it seems that CGAs are very bad at solving these problems.

SGA:
These work very poorly if using uniform crossover, I would even say they work about as well as the CGAs. with one point crossover they work very well though. Much better than CGAs though. .

I would say that CGA is much worse than any form of SGA in relation to trap problems. If you think about how CGAs work with no population, that makes it much harder for them to have any good trap segments and then it makes it hard for the model to “remember” any good trap segment they get.

For both of these it’s also difficult to compare because of how differently they work. For previous projects we could just compare generations, but here it might be more apt to compare runtime. Most examples here were sufficiently close.