ArithmeticPuzzle Writeup

I evolved a solution using an array of digits, its length equal to the number of unique letters in the input, where each digit was in the range 0…9.

I created a random instance of the population by selecting an independently random digit for each index of the array.

At the beginning of my program, I created an array of all letters appearing in the problem. To determine the fitness of a chromosome I first reinterpretted each duplicate digit in its array as the next highest digit not already in the array. I converted each input String into a number by checking the index of each letter, and interpreting the number at that index in the chromosome as its value. I defined fitness as the absolute value of the equation “augend + addend – sum”, where a smaller number was more fit, and the threshold as that equation equalling 0.

My implementation performed mutation by selecting a random index in the given chromosome’s array. It replaced the digit at that index with a random digit within the range 0…9.

My implementation performed crossover by selecting a random index on the given chromosomes’ arrays. For each index starting at 0, I swapped the digit on one chromosome for the digit on the other chromosome until I reached the selected index.

To determine the best selection type and crossover and mutation probabilities, I initially ran 4 series of trials: one in Tournament mode, experimenting with mutation, and controlling crossover; another in Tournament mode, experimenting with crossover, and controlling mutation; and two other experiments identical to the first two but in Roulette mode. I had assumed that crossover and mutation rate were independent variables. The controlled probability was set to 0.5. The experimental probability was tested every 0.1-precision double from 0.0 (for crossover) or 0.1 (for mutation) to 1.0. I ran each trial using a population of 1000 over 100 generations. The augend, addend, and sum were each length-10 numbers containing between them 9 distinct digits. I ran each trial 50 times, averaging the number of generations it took to find a correct answer.

I found that while both selection modes were fine for smaller modes, for larger modes, Tournament mode was much better than Roulette mode. Tournament mode, on average, took 71 generations to calculate the solution, while Roulette took 97 (generally failing on an input this large). Optimal mutation was 0.9, averaging 58 generations. Optimal crossover was 1.0, averaging 58 generations. But running with all 3 variables set this way, I found an average of 88 generations, nearly always failing. This suggested that mutation and crossover chance are not independent variables.

Instead, I ran every combination of 0.1-precision mutation and crossover probabilities in Tournament mode (since I had found it to be superior to Roulette). This way, I found that the best setting was a mutation probability of 0.7 and a crossover probability of 0.8, which averaged 50 generations.