DataCompression

Isaac Gutt

April 2022

1

I created a new copy of the original list and then used Collections.shuffle() to get a random ordering of the list.

2

The relative improvement method, the higher it is the better the solution.

3

Depending on the list, I'd run tests to see how high improvement could go and put the threshold at that point, the one I used for my list was 1.31.

4

In each generation, I ran a loop through the chromosomes and picked a random double from 0-1, if it was lower than mutation chance, the solution would get mutated. To mutate, I took two random indexes within the population size and swapped the strings from their respective indexes to the others.

5

If crossover happened, I created copies of the two parents selected for crossover, and then filled a new list of size/2 with random strings in the list, if the strings appeared more than once in the list, I filled another list with the indexes that it appeared at. Then, for each string in the selections, childA would have that string moved to the index where childB had it and vice versa, if there were multiple indexes the string was at, then one would be picked at random.