Data Compression

Jonathan Wenger

1. I created a random instance by calling Collections.shuffle on the original list, and adding that to the population.
2. My fitness criterion was just calling relativeImprovement on the instance; the greater the relative improvement, the better fit chromosome it is.
3. My threshold was bytesCompressed(original list). There's no official "threshold" we need to pass, so I set the threshold insanely high (as the fitness of any individual chromosome cannot get close to bytesCompressed(original list), but will never be greater than that value).
4. After running thousands of tests to see which selection type produced the best result, I do not have a conclusive answer. Each selection type performed well, but I did not find that one selection type produced better results than the other.
5. The way my implementation performed mutation was that it took a few pairs of random indices in the list, and swapped the strings at those indices with each other.
6. The way my implementation performed crossover was it took two chromosomes and made one new chromosome out of them. I iterated through both chromosomes, adding all the odd-indexed strings of A and even-indexed strings of B, and any overlap/missing Strings were dealt with.