ICS674 Mini Project

Lambert Leong

1 Search Space

Listing 1: Search space is all letter characters, upper and lower case

```
self.string = ''.join(random.choice(string.letters) for _ in
    xrange(length))
```

2 Variation Operator

2.1 CrossOver

```
def crossover(individuals):
        offspring = []
        for _ in xrange((population - len(individuals))/2):
                parent1 = random.choice(individuals)
                parent2 = random.choice(individuals)
                child1 = Individual(in_str_len)
                child2 = Individual(in_str_len)
                split = random.randint(0, in_str_len)
                child1.string = parent1.string[0:split] +
                    parent2.string[split:in_str_len]
                child2.string = parent2.string[0:split] +
                    parent1.string[split:in_str_len]
                offspring.append(child1)
                offspring.append(child2)
        individuals.extend(offspring)
        return individuals
```

2.2 Mutation

```
def mutation(individuals):
        for individual in individuals:
```

```
        for i, param in enumerate(individual.string):
            if random.uniform(0.0, 1.0) <= 0.05:
                individual.string =
                    individual.string[0:i] +
                    random.choice(string.letters) +
                    individual.string[i+1:in_str_len]
    return individuals
```

3 Selection Operator

```
def selection(individuals):
    individuals = sorted(individuals, key=lambda individual:
        individual.fitness, reverse=True)
    max_fit.append(max(individuals, key=lambda individual:
        individual.fitness).fitness)
    min_fit.append(min(individuals, key=lambda individual:
        individual.fitness).fitness)
    avg_fit.append(float(sum(i.fitness for i in
        individuals)//len(individuals)))
    individuals = individuals[:int(0.2*len(individuals))]
    return individuals
```

4 Termination Criterion

```
 for generation in xrange(generations):
        generation_list.append(generation)
        individuals = fitness(individuals)
        individuals = selection(individuals)
        individuals = crossover(individuals)
        individuals = mutation(individuals)
        if any(individual.fitness >= 100 for individual in
            individuals):
            found = True
            break
```

5 Objective Fuction

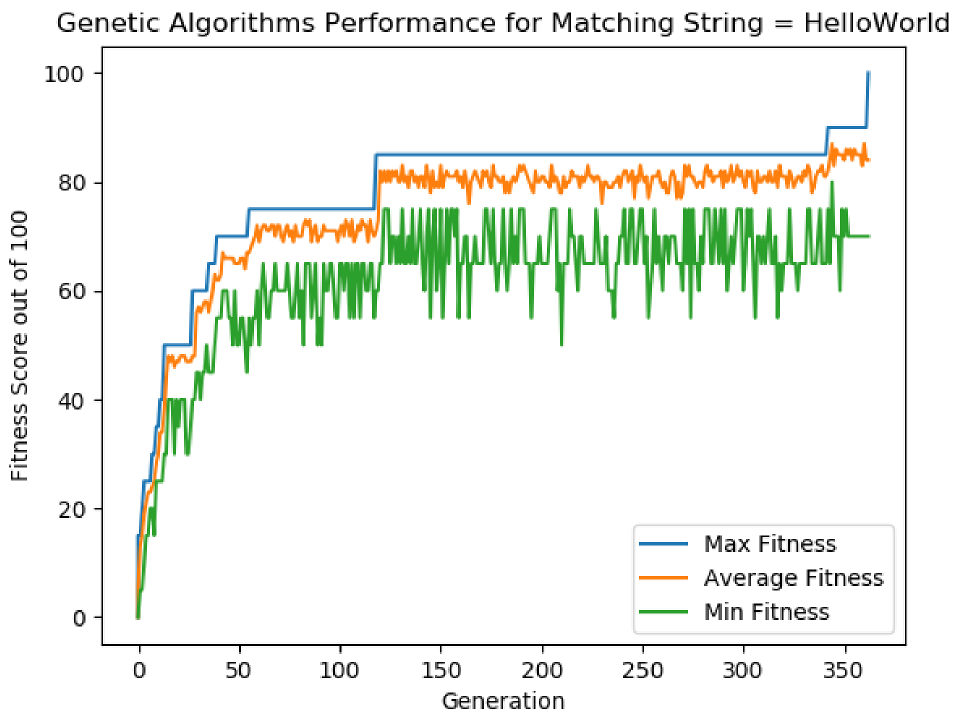

Figure 1: Max, average, and min fitness values of each indvidual string for each generation

Listing 3: Fitness Function

```
def fitness(individuals):
        for individual in individuals:
                total = len(in_str)*2
                score = 0
                for i, letter in enumerate(individual.string):
                                if in_str[i] == letter:
                                        score += 1
                compare_str = in_str
                for a_char in individual.string:
```

```
            for i, in_char in enumerate(compare_str):
                if a_char == in_char:
                    score += 1
                    compare_str =
                      compare_str[:i]+compare_str[i+1:]
                    break
        individual.fitness =
          int((float(score)/float(total))*100)
    return individuals
```