Genetic Algorithm Assignment #1

B0591025 Yu-Neng Wang

November 1, 2019

Instructor: Prof. Tian-Li Yu

Problem

1. **Let's check how often deception occurs in a random function. The definition of deception can be found in the lecture slides**.

   (a) **Consider the case with two genes. By assigning fitness values for f(00), f(01), f(10), and f(11), does deception ever occur? Prove it cannot happen if not, give an example if yes.**

   No, deception never occurs.
   Proof: Without loss of generality, assume the global maximum is $f(11)$, In order to make deception occur, the fitness function should satisfy

   $$\begin{cases} f(0*) = f(00) + f(01) > f(1*) = f(10) + f(11) \\ f(*0) = f(00) + f(10) > f(*1) = f(01) + f(11) \end{cases}$$

   Substitute the lower equation into upper one, we get

   $$f(00) + f(01) > f(01) + f(11) - f(00) + f(11) \Rightarrow f(00) > f(11)$$

   Contradiction occurs because $f(11)$ should be the largest. Hence, deception won't happen in the case with two genes. *Q.E.D.*

   (b) **Consider the case with three genes, randomly assign the fitness values for $f(000), f(001), f(010), f(011), f(100), f(101), f(110), f(111)$ with uniform distribution from 0 to 1. Repeat the experiments 10^6 times. What's the probability that 3-deception occurs?**

   0.005693

   (c) **Repeat (b), but with 4 genes.**

   0.000318

   (d) **For a problem with *ell* genes (problem size), the probability that *k*-deception does *NOT* occur among any *k* genes is roughly $(1-p)^{c_k^{ell}}$, where *p* is what you record in (b) and (c). What's the problem size that makes 3-deception occur with probability 0.5? What's that for 4-deception? When does 3-deception occur more often than 4-deception or the other way around? Write a short essay of your finding.**

Solving ell for

$$\begin{aligned} & 1-(1-p)^{C_k^{ell}} = 0.5 \\ \Rightarrow\ & C_k^{ell}\log(1-p) = \log(0.5) \\ \Rightarrow\ & ell(ell-1)...(ell-k+1) = k!\frac{\log(0.5)}{\log(1-p)} \end{aligned}$$

we can get

$$\begin{aligned} k=3, p=0.005693 \Rightarrow ell = 10.035 \\ k=4, p=0.000318 \Rightarrow ell = 16.664 \end{aligned}$$

Compare $1-(1-0.005693)^{C_3^{ell}}$ with $1-(1-000318)^{C_4^{ell}}$, because C_4^{ell} has higher order than C_3^{ell}, the latter's $(1-p)$ term will decade faster. Hence, 4-deception will occur more often than 3-deception when:

$$\begin{aligned} & 1-(1-0.005693)^{C_3^{ell}} \leq 1-(1-000318)^{C_4^{ell}} \\ \Rightarrow\ & C_3^{ell}\log\left(0.99431\right) \geq C_4^{ell}\log\left(0.99968\right) \\ \Rightarrow\ & \frac{ell(ell-1)(ell-2)}{3!}\log\left(0.99431\right) \geq \frac{ell(ell-1)(ell-2)(ell-3)}{4!}\log\left(0.99968\right) \\ \Rightarrow\ & ell \geq 75 \end{aligned}$$

From the above experiment, I got 2 conclusions. First, For one assignment of fitness, k-deception has lower chance to occur than lower order deception. However, when consider k-deception in one chromosome with length ell, we need to consider all C^ell_k combination, which leads to higher order (ell^k) in exponent. Hence, the probability of the occurrence of k-deception will eventually be higher than lower order ones.
Second, because the probability that k-deception does not occur among k genes decays exponentially, the probability that k-deception occur approximate to 1 even when ell is not large. Before I run this experiment, I thought that deception is hard to occur (Therefore I do not understand why we discuss deception a lot in class). Now I know that deception occurs very often, and that is why we need to discuss about how epsilon affect convergence, building block and those theory in genetic algorithm.

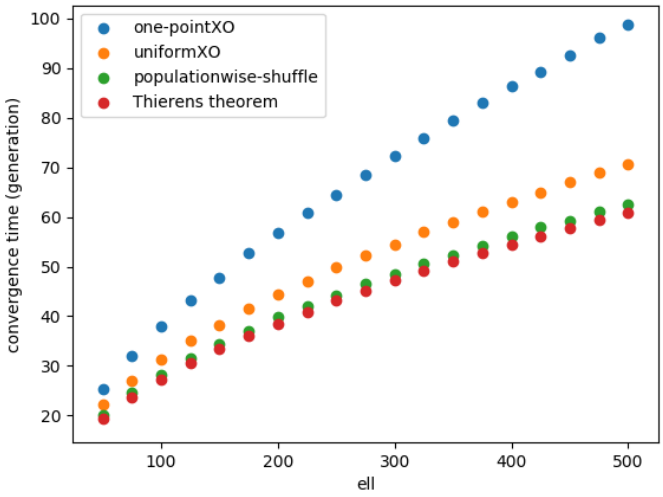
2. **Thierens' convergence-time model assumes perfect mixing. Lets verify the model with one-point XO, uniform XO, and population-wise shuffling. In population-wise shuffling, the set of genes at a particular position of parents. (Use tournament selection of size 2 without replacement and no mutation.)**

   (a) **Experiment these three XO's with SGA on the OneMax problem with different sizes (*ell*) of 50, 100, 150, 200, 250, 300, 350, 400, 450, 500. Plot the results on a figure with problem size versus convergence time. (You need to average at least 30 independent runs to get stable results. Population size should be set at $4 * ell * ln(ell)$.)**

   I set the experiment parameter to be chromosome_length=ell, population_size=$4*ell*int(ln(ell)+1)$, selection_intensity=2, crossover_probability=1, mutation_probability=0, and repeat 50 times, i.e. in the sample code format:

   $$./sga\ ell\ 4 * ell * int(ln(ell) + 1)\ 2\ 0.5\ 0\ -1\ -1\ 50$$

   After record and average the output, I plot the figure. Further discusion will be in 2.-(c).

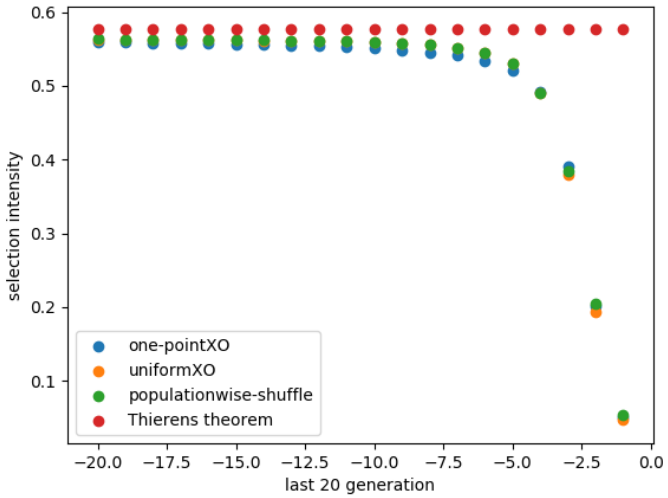


   (b) **What's the theoretical value of selection intensity? How does that compare with the selection intensity measured in your experiment? Is selection intensity really invariant with generation?**

   First, compute the selection intensity I, when $s_t = 2$

   $$\begin{aligned} I &\cong \sqrt{2(\ln s_t - \ln \sqrt{4.14 \ln s_t})} \\ &= 0.576 \end{aligned}$$

The theoretical value is very close to the selection intensity measured in my experiment and almost invariant with generation. As shown below, the selection intensity keeps the same until about last 10 generation. The drop of selection intensity of last 10 generation also makes sense because that's the time when chromosome start to converge, and lead to smaller diversity of population (being dominated by the best solution).



(c) **How does Thierens' model compare with your results? Write a short essay of your finding.**

We can compute t_{conv} in Thierens' model using the following equation:

$$t_{conv} = \frac{\pi}{2}\frac{\sqrt{l}}{I}$$

From the above experiment and compare the red-dot with green-dot in 2-(a), the "populationwise-shuffle" behaves the most alike to Thierens' model, following by uniformXO, and then one-pointXO. In my opinion, this is because Thierens' model considers "perfect mixing", populationwise-shuffle mixes the genes based on whole pipulation, which is more completely and randomly. Meanwhile, uniformXO and one-pointXO only mixes the genes sampled from 2 chromosomes and one-pointXO also be affect by the physical location of the genes (neighboring genes has a higher chance to remain in the neighbor than distanced genes). Therefore, the latter 2 XO is not "perfect" enough, so their t_{conv} are a little away from theoretical value. Also, I find that if I set the XO's probability to 0.5, the one-pointXO might fail sometimes due to convergence to non-optimum solution, while the other 2 XO don't. I think this is because one-pointXO converges so slow that drift will takeover the population even in this simple task.