Introduction to evolutionary algorithms (chapter 7)

Let I be the space of individuals, such that, a ∈ I, and F: I → R be the fitness function.

Using μ and λ to denote parent (old generation) and offspring (new generation) population sizes, P (t) = (a1 (t), . . . , aμ (t)) ∈ Iμ characterizes a population at generation t.

Parameters symbols:

* Selection: s
* Mutation: m
* Recombination: r

Algorithm:

1. t ← 0;
2. P (t) ← initialize(μ);
3. F (t) ← evaluate(P (t), μ);
4. while (true) do
5. P’ (t) ← recombine(P (t), r );
6. P’’ (t) ← mutate(P’(t), m );
7. F (t) ← evaluate(P’’(t), λ);
8. P (t + 1) ← select(P’’(t), F (t), μ, s );
9. t ← t + 1;

* While P(t) consist of μ individuals, P’(t) and P’’(t) consist of k and λ, respectively. But, they can be the same value (default for a lot of algorithms).

Genetic algorithms (chapter 8)

Selection

Most Genetic algorithms maintain the same amount of population (M). Let’s say that the number of offspring is higher than M. What can be done is that we can choose the number of individuals from the parents to complete the offspring population. This can be done by choosing by random, or choosing the best ones (better). If the number of offspring generated is M, then all parents are killed (usually you want to maintain the best one).

Fitness Differences:

A way to calculate the probability of an individual being selected is to sum up all the fitness into the total fitness and then, for each individual, divide it’s fitness by the total, but this approach has a problem. Let’s consider functions y = ax² and y’ = ax² + b, they are similar, but the fitness differences on both can be huge.

Imagine that the b constant is a lot bigger than the result of ax², this means that, for all x, y’ will have almost the same value, therefore, almost the same fitness, making it difficult to know which individual to select.

To solve this problem we can implement a type of calculation called scaling fitness. This approach takes into account the probability according to the worst one.

Scaling fitness also has its problems. Although it eliminates the problem of not enough selection pressure it causes a lot of pressure, that is, if an individual is much better than the other ones, specially when compared to the worst one, it has a high probability that he will take over the population, decreasing the diversity. There are other types of methods to avoid this:

Ranked:

* The individuals in the parent population are ranked, and the probability of selection is a linear function of rank rather than fitness, where the ‘steepness’ of this function is an adjustable parameter.

Tournament selection:

* A small number of individuals is chosen at random, and the best individual (or two) are selected for the mating buffer (breeding).
* With this method, a single individual is not capable of taking over a population that fast.
* Not on the book, but, I think that, beside choosing the subset at random, it is also important to maintain the best parent at the offspring population, even if he was not selected

The main difference between having one or two mating per cycle versus many matings, is that:

* with less mating per cycle, the offspring population can vary widely in terms of their fitness. This variance can potentially lead to more diverse solutions being explored at the same time that adds uncertainty to the problem.
* With more mating per cycle, the offspring tends to converge into an optimal solution

Replacing the parents by the children without hesitation can lead to losing, ***perhaps forever***, very good individuals.

Mutation and crossover

One-point crossover:

* a single locus is chosen at random and all bits after that point is swapped.

Two-point crossover:

* Two points are chosen at random and swap the genes of these two points between the parents.

Uniform crossover:

* Randomly swaps individual bits between the parents.