GENETIC ALGORITHMS

We will study a simple example, that of deciding the optimal way to paint a number of windows NW using NCOL different colors (for example, for 4 windows and a choice of 4 different colors, one of many possible ways to paint these windows is 1432, where 1 stands for BLUE, 2 for YELLOW, 3 for RED, 4 for GREEN).
As we explained in class, there are the following phases in a genetic algorithm solution of an optimization problem:

1. First decide on a sequence of "genes" that describe a solution, such as the one we gave above, 1432. For all the convergence properties to hold, we ought to use dyadic notation (0 and 1) so this is a deviation from the formal procedure. Had we decided to go the formal way, BLUE would be 00, YELLOW 01, RED 10, and GREEN 11, hence the sequence 1432 would become 00111001, i.e. a longer sequence.
2. Second, we decide on a number for the population size NPOP. The genes of each population sample are randomly selected.
3. Third, we rank each sample, using an **evaluator**. This means there is a way to place a "grade" on each sequence of genes. In our example, let's say the ideal combination is BLUE, BLUE, YELLOW, YELLOW, or 1122: Then a choice of 1432 would get, for example, a grade of 1 because only one window is "correctly" chosen.
4. Now is the time for **gene-crossing**. We select two samples of the population (the parents) – they can be two samples of the population chosen randomly, or we may choose the two highest ranking ones, then the two next highest ranking -- or any variation in between such methods. Next, randomly "exchange" the genes of the two parents. For example, with a random number generator we select the first gene to be from parent one if the number is less than 0.5, otherwise it is from the second parent; continue for the second gene, and so on. This process generates a new sample, an offspring. We generate several offsprings.
5. Next is time for **mutation**. First we choose a probability of mutation PMUT: For each gene of every offspring we decide with a random number generator whether we will mutate or not. If we mutate, then we exchange that particular gene randomly with another one (if it is BLUE we mutate it randomly to RED, or GREEN, or YELLOW).
6. Next we use the evaluator for the offsprings to rank them, and put together parents and offsprings to form the new population. We throw away the bottom performers out of the group and retain the top NPOP samples (from the combined group of parents and offsprings). Next we go back to step 4 and repeat until we decide that we have converged.

Convergence is usually defined when a large number of the population samples have the same genes and a high grade (Say half the population). It is possible to have reached a local maximum – if we are not sure, for example, what a perfect score is (in our example we do, but there are cases when the evaluator gives relative scores only). Mutation is a way to move out of a local maximum, so it may pay to use, at least initially, a high probability of mutation, but it may also slow down convergence by moving solutions randomly around.

EXAMPLE 1: We have four windows (NWIND=4), 3 colors (NCOL=3), a population size of eight (NPOP=8), and a probability of mutation 10% (PMUT=0.1). The ideal combination is: 2132.

Only four new offsprings are produced in each gene-crossing and mutation iteration. As an example, let's see the gene crossing and mutation in iteration 1 out of a population of eight, ranked as 3 2 2 2 1 0 0 0:

(a) Select as parents the first and third (in order of ranking):

Genes of first:	2	1	2	2
Genes of second:	2	1	1	1
Offspring	2	1	2	1

(b) Next select the second and fourth:

Genes of second	1	1	1	2
Genes of fourth	2	2	3	3
Offspring	1	2	3	2

And so on (since the probability of mutation is low, none of the genes above was mutated).

The progress of convergence is as follows:

ITERATION	HIGHEST GRADE	MID-POPULATION GRADE	# OF SAMPLES WITH PERFECT GRADE
1	3	2	0
2	3	2	0
3	3	2	0
4	3	3	0
5	3	3	0
6	3	3	0
7	3	3	0
8	3	3	0
9	3	3	0
10	4	3	1
11	4	3	1
12	4	3	2
13	4	3	3
14	4	4	4
15	4	4	4

Convergence has been achieved practically at the 14th iteration. Four new offsprings are generated in each iteration, hence 14*4=56 evaluations were made. With an exhaustive search, 3^4=81 evaluations would be needed, so this is not a complex enough problem for genetic algorithms.

EXAMPLE 2: Next we consider a more complex problem with 10 windows (NWIND=10) and a choice of 4 colors (NCOL=4) for each windw, and mutation probability of 10%. The ideal combination is: 2132441124. Here is how the solution evolves with a population of 50 (NPOP=50):

ITERATION	HIGHEST GRADE	MID-POPULATION GRADE	# OF SAMPLES WITH PERFECT GRADE
1	6	2	0
2	6	3	0
3	7	4	0
4	7	5	0
5	7	6	0
6	8	6	0
7	9	7	0
8	9	7	0
9	9	7	0
10	10	8	1
11	10	8	1
12	10	8	2
13	10	9	3
14	10	9	5
15	10	9	9
16	10	9	13
17	10	9	22
18	10	10	25

The algorithm has converged after 18 iterations, when half the population has a perfect score of 10. In each iteration 25 new offsprings are generated (half the population) requiring for convergence an evaluation of 18*25=450 samples; an exhaustive search would require 4^{10}=1,048,576 evaluations. Here genetic algorithms have a clear advantage, by a factor of at least 2,000.

DISCUSSION

The choice of population size, mutation probability (which may change from a high value in the beginning to a low value as iterations progress), and algorithm for choosing parents, affect the success and speed of convergence of genetic algorithms so experimentation is usually needed.