* What type of problem can be solved using genetic algorithm?

A. GAs are not good for all kinds of problems. They’re best for problems where there is a clear way to evaluate fitness. If your search space is not well constrained or your evaluation process is computationally expensive, GAs may not find solutions in a sane amount of time. In my experience, they’re most helpful when there is a decent algorithm in place, but the “knobs” just need to be tweaked.

* Give example of at least three problem from different field of life solved by using genetic algorithm?

### 1. Automotive Design

### 2. Engineering Design

### 3. Robotics

### 4. Evolvable Hardware

### 5. Optimized Telecommunications Routing

### 6. Joke and Pun Generation

### 7. Biomimetic Invention

### 8. Trip, Traffic and Shipment Routing

### 9. Computer Gaming

### 10. Encryption and Code Breaking

* Give Cleary identify the initial population representation, evaluation, function, mutation and exit criteria?

### Genetic algorithms (GAs) may contain a chromosome, a gene, set of population, fitness, fitness function, breeding, mutation and selection. Genetic algorithms (GAs) begin with a set of solutions represented by chromosomes, called population. ... Better the fitness, the bigger chance to be selected to be the parent.

### The Algorithm

### In the genetic algorithm process is as follows [1]:

### Step 1. Determine the number of chromosomes, generation, and mutation rate and crossover rate value.

### Step 2. Generate chromosome-chromosome number of the population, and the initialization value of the genes chromosome-chromosome with a random value

### Step 3. Process steps 4-7 until the number of generations is met

### Step 4. Evaluation of fitness value of chromosomes by calculating objective function

### Step 5. Chromosomes selection

### Step 6. Crossover

### Step 7. Mutation

### Step 8. Solution (Best Chromosomes)

### as_ga_algorithm.gif

### Initial Population

As described above, a gene is a string of bits. The initial population of genes (bitstrings) is usually created randomly. The length of the bitstring is depending on the problem to be solved.

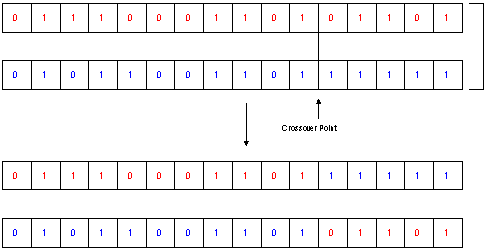
### Selection

**Selection** means to extract a subset of genes from an existing (in the first step, from the initial -) population, according to any definition of quality. In fact, every gene must have a meaning, so one can derive any kind of a quality measurement from it - a "value". Following this quality "value" (fitness), Selection can be performed e.g. by Selection proportional to fitness:

1. Consider the population being rated, that means: each gene has a related fitness. (How to obtain this fitness will be explained in the [Application](http://www.ifs.tuwien.ac.at/~aschatt/info/ga/genetic.html#ApplicationSection)-section. ) The higher the value of the fitness, the better.
2. The mean-fitness of the population will be calculated.
3. Every individuum (=gene) will be copied as often to the new population, the better it fitness is, compared to the average fitness. E.g.: the average fitness is 5.76, the fitness of one individuum is 20.21. This individuum will be copied 3 times. All genes with a fitness at the average and below will be removed.
4. Following this steps, one can prove, that in many cases the new population will be a little smaller, than the old one. So the new population will be filled up with randomly chosen individua from the old population to the size of the old one.

**Remember, that there are a lot of different implementations of these algorithms.** For example the Selection module is not always creating constant population sizes. In some implementations the size of the population in dynamic. Furthermore, there exist a lot of other types of selection algorithms (the most important ones are: Proportional Fitness, Binary Tournament, Rank Based). I restrict myself to describe just the most common implementations in this short article. To get a deeper insight to this topic take a look to the [Recommended Reading](http://www.ifs.tuwien.ac.at/~aschatt/info/ga/genetic.html#RecommendedReadingSection) section.

### Mating/Crossover



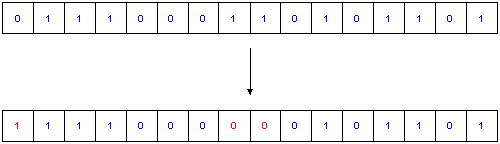
The next steps in creating a new population are the **Mating**and **Crossover**: As described in the previous section there exist also a lot of different types of Mating/Crossover. One easy to understand type is the random mating with a defined probability and the b\_nX crossover type. This type is described most often, as the parallel to the Crossing Over in genetics is evident:

1. PM percent of the individua of the new population will be selected randomly and mated in pairs.
2. A crossover point (see fig.2) will be chosen for each pair
3. The information after the crossover-point will be exchanged between the two individua of each pair.

In fact, more often a slightly different algorithm called b\_uX is used. This crossover type usually offers higher performance in the search.

1. PM percent of the individua of the new population will be selected randomly and mated in pairs.
2. With the probability PC, two bits in the same position will be exchanged between the two individua. Thus not only **one** crossover point is chosen, but each bit has a certain probability to get exchanged with its counterpart in the other gene. (This is called the uniform operator)

### Mutation



The last step is the **Mutation**, with the sense of adding some effect of exploration of the phase-space to the algorithm. The implementation of Mutation is - compared to the other modules - fairly trivial: Each bit in every gene has a defined Probability P to get inverted.