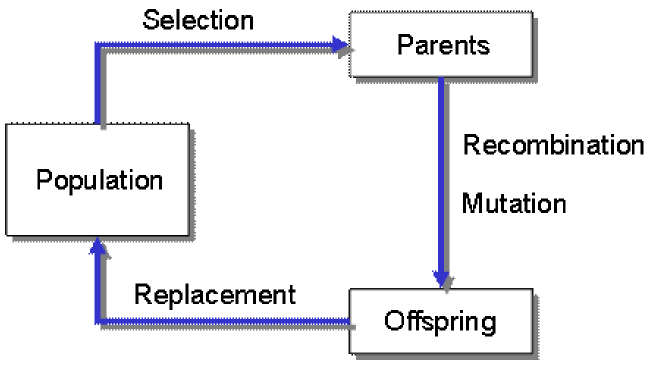
Evolutionary Search

Finally, we will consider a whole class of stochastic search algorithms, referred to by several names but falling under the general category of evolutionary search. We will focus on one particular example of evolutionary search called the genetic algorithm.

Genetic algorithms are based on the biological principle of "survival of the fittest." The approach uses a number of other ideas from genetics and evolutionary biology (as well as a lot of the terminology). For example, candidate solutions are encoded into "chromosomes." In addition, the search process maintains a population of candidates rather than focusing on a single candidate solution. The process then follows the evolutionary cycle of allowing the individuals in the population to "reproduce" and generate new individuals. Selection who reproduces and who survives is based on the fitness function. This process is illustrated as follows:



The prototypical genetic algorithm is represented in the following pseudo-code.

Algorithm 1 Prototypical Genetic Algorithm

```
GeneticAlgorithm
  $t \leftarrow 0$
  Initialize($P(t)$)
  EvaluateFitness($P(t), f(t)$)
  while (not terminatep()) do
    $t \leftarrow t + 1$
    $C(t) \leftarrow$ Select($P(t-1)$)
    $C'(t) \leftarrow$ Recombine($C(t)$)
    $C''(t) \leftarrow$ Mutate($C'(t)$)
    EvaluateFitness($C''(t), f(t)$)
    $P(t) \leftarrow$ Replace($C''(t), P(t-1)$)
  end do
```

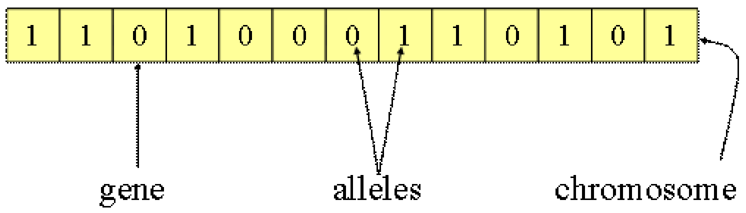
As shown, the first step is to generate a population of individuals at random. Each of these individuals is evaluated according to the fitness function, and the evolutionary cycle begins. First, a group of individuals is selected from the population (at random) and "recombined" to produce offspring individuals. To introduce another level of randomness, a mutation operator is also applied to make small variations in some of the offspring. Next the offspring are evaluated using the fitness function, and a replacement procedure is used to determine which individuals out of the whole set (original population + offspring) survive to the next generation.

Issues in Evolutionary Search

There are several issues (and related design decisions) that need to be addressed when applying this approach. These issues include:

1. What is the appropriate representation for the chromosome?
2. How is fitness supposed to be measured?
3. How does one select individuals from the population?
4. How are these individuals "recombined" to generate offspring?
5. How do we decide who gets replaced in the population?

In the so-called "standard" genetic algorithm, all individuals are represented as bit-strings. Each bit (or subsequence of bits) corresponds to an attribute of the problem and are called "genes." The specific values of the genes are called "alleles." The entire sequence is the chromosome.



Of course, other representations are possible, including real numbers, symbolic strings, trees, graphs, and even programs. In fact, the version of the genetic algorithm that works on program parse trees is called genetic programming.

Determining the fitness is tied to the problem being solved. In many cases, fitness can be computed directly from the problem definitions, but sometimes, more sophisticated fitness functions are required. The evolutionary computation community distinguishes between two types of fitness functions—exogenous fitness and endogenous fitness. Exogenous fitness involves some "external" evaluation of an individual, where endogenous fitness has the individual directly competing with other individuals for resources built in to the environment. Ultimate, the goal of fitness evaluation is to determine the relative worth of the individuals in the population.

Given we can compute the relative fitness of the individuals, several approaches to selection suggest themselves. One is called "fitness proportional selection." In this approach, probability of selected is computed based on the relative fitness of the individuals to the total fitness of the population.

$$P(sel = x_i) = \frac{f(x_i)}{\sum_{j=1}^{|P|} f(x_j)}.$$

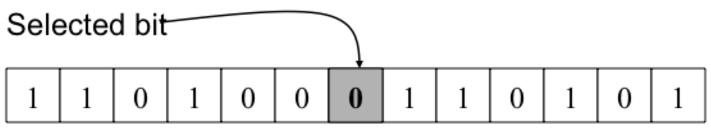
Selection is done with replacement, so it is possible for an individual to be selected more than once.

Another approach is called "tournament selection" where individuals complete in head-to-head tournaments based on their fitness value. In a tournament of size q, q individuals are selected at random according to a uniform distribution, and the best individual (based on fitness) in the tournament is the one selected. Generally, $q = 2$.

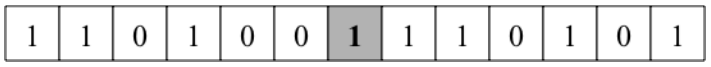
Finally, replacement can be done in a way similar to selection. Key to the replacement process is determining how much of the original population to replace. If the entire population is replaced, the replacement process is called "generational." On the other hand, if only a subset is replaced, then replacement is said to be "steady-state." It is common for steady-state replacement to be used where only two individuals are replaced in a generation.

Genetic Operators

The other issue to be considered is how individuals are recombined to generate offspring. Several "operators" have been suggested to do this; however, we consider the two most common here. First, mutation is the process of randomly altering a portion of an individual. For example, suppose we have a binary chromosome and select one of the bits for mutation.

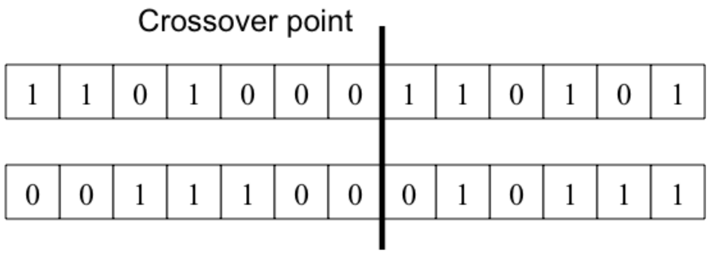


Since we are dealing with a binary chromosome, mutation results in that bit being flipped.

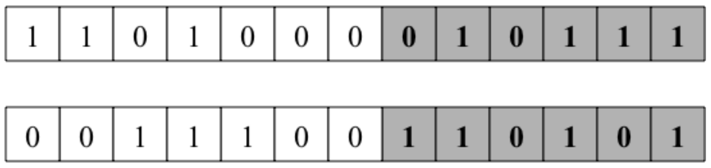


Mutation only makes changes to single individuals in the population. When generating offspring, it is desirable to generate individuals that share “genetic material” from two or more parents. The crossover operator is the one most generally used for this.

In crossover, two individuals are selected, and (for one-point crossover) a crossover point is selected at random.



Once the crossover point is selected, two offspring are generated where the right had sides of the original two individuals are swapped.



Why Does Evolutionary Search Work?

Several theories have been suggested to explain why the evolutionary search approach works. One of the oldest and more popular explanations builds on what is called schema theory. In the genetic algorithm, a schema can be thought of as a template for matching chromosomes. To define these templates, an extra character, corresponding to a "don’t care" is used. It is common for “#” to be used as the don’t care symbol. For example, the schema [#01####1] would match any chromosome with a zero in the second position and ones in the third and last position. [10100111] is an example instance of a chromosome matching this schema.

Schemas are sometimes called building blocks in genetic algorithms, and David Goldberg proposed the so-called “Building Block Hypothesis.” According to Goldberg, a genetic algorithm seeks near optimal performance through a juxtaposition of short, low-order, high-performance schemata, called building blocks. This hypothesis has not been proven and has only been demonstrated in limited settings.

Two fundamental results that arise from schema theory are the *schema theorem* and a concept called *implicit parallelism.* Starting with the latter, implicit parallelism says that GAs process $\gg n$ schemata at every generation (for a population of size n). The schema theorem (which we will consider in more detail in a moment), says that better templates dominate the population as the generations proceed. The schema theorem and associated theory are controversial, and alternative theories based on Markov processes are being offered.

Theorem (Schema): Short, low-order, above average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm. In other words,

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{\bar{f}}\left[1 - p_c\frac{\delta(H)}{\ell - 1} - o(H)p_m\right].$$

Proof Sketch: The full proof, though not difficult, is beyond the scope of this course, so we summarize the proof here. First, some terms need to be defined. Let H represent a particular schema, and let $m(H,t)$ represent the number of individuals in the population at time t that match schema H. $f(H)$ is the fitness of the schema as measured by the average fitness of all of the individuals in the population matching H. $\bar{f}$ is the average fitness of all of the individuals in the population. The parameters p_c and p_m represent the probability of crossover and mutation respectively. Finally, $o(H)$ represents the "order" of schema H (i.e., the number of non-don't-cares in the schema), and $\delta(H)$ represents the "defining length" of schema H (i.e., the number of bits between the first and last non-don't-care bits, inclusive). We use ℓ to denote the length of the schema.

When looking at this "schema growth equation" in the theorem, note that $\left[1 - p_c\frac{\delta(H)}{\ell-1} - o(H)P_c\right]$ is constant for a particular schema H. This means it has no long-term effect on schema growth (similar to the constants in the asymptotic equations), so we can ignore this term. This means we need to consider $m(H,t+1) \geq Km(H,t)f(H)/\bar{f}$ where K is the above constant. Of note is the fact that, when $f(H) > \bar{f}$, we will tend to see the number of instances matching H increase over time. They will decrease over time if $f(H) < \bar{f}$. This means we can represent schema growth as a geometric progression as follows:

$$m(H,t) = m(H,0)\left(1 + \frac{f(H) - \bar{f}}{\bar{f}}\right)^t.$$

This geometric progression completes the proof. QED