## 0.1 Genetic Algorithms

Genetic Algorithms (GA) are a class a meta-heuristic search algorithms used for many different types of optimization problems. The main idea behind these algorithms is that they employ a search strategy similar to that of biological evolution and natural selection. This chapter firstly introduces the theory behind GAs and consequently describes how they have been used within the domain of counterpoint generation.

Being an algorithm inspired by biological phenomena, the theory of GAs has adopted terms from the field of genetics. When defining a GA one first needs to specify what the *genotype* and *phenotype* are. Within biology the genotype refers to the genetic makeup which effectively *encodes* a physical trait, or phenotype. In the case of GA, the distinction is analogous. The genotype is an effective encoding of what the algorithm is optimizing. For the problem of counterpoint generation, one could view the genotype as a sequence of pitches, which represents the phenotype, a phrase of music. Therefore, given a genotype representation, the space of possible genotypes is the search space. An objective or fitness function is a function that takes a genotype and outputs a genotype's fitness, i.e, a measure of how good it is. The idea is thus to find a genotype that maximises the objective function. Genetic Algorithms attempt to find these by using the following procedure.

- **Initialization**
  A set of $N$ random genotypes are generated, representing the current *population*. This population represents the first *generation* of candidates.
- **Evaluation**
  The current population is evaluated according to the objective function.
- **Selection**
  A selection procedure is employed in order to select $k$ individuals from the current population. Typically the selection procedure randomly selects an individual proportional to their fitness value. Another possible method is to just select the $k$ best individuals.
- **Cross-over**
  $N$ new individuals are created by recombining the genotypes of two random individuals selected from the $k$ best. The type of recombination is highly domain dependant, although the idea is to be able to retain and combine the information of the two individuals so that the new genotype has an equal or better fitness.
- **Mutation**
  For each of the $N$ new individuals, their genotype is mutated by some probability $\alpha$. This mutation usually entails changing a small part of the genotype. For example, if a binary representation is used, mutation would change the value of a bit with probability $\alpha$. Mutations are used in order to try to prevent the algorithm from prematurely converging on local maxima(**Undefined reference**). Higher values of $\alpha$ will result in more exploration of the search space although it will take longer for the problem to converge at a optimal solution (**Undefined reference**).
- **Repeat**
  The algorithm returns to the evaluation phase and the process continues until some stopping criteria is reached. Again, this criteria is highly domain dependant. As an example, one could stop searching when there is no change between the maximum fitness value of two generations.

Genetic Algorithms have seen a number of sucesses in various applications. In particular, they have become a popular go to method for various problems algorithmic composition ().
**TODO: add more information about literature of GA successes within the domain of music composition**.
In order to determine how well the Monte Carlo based approach fares within the domain, it was tested against a Genetic Algorithms within the domain of first species counterpoint generation. To this end a Genetic Algorithm similar to ones found in the literature was defined. The specifics of the algorithm are found in a subsequent chapter.