# OptimizeArmBracket1

**OptimizeArmBracket** is a C# program that designs an arm bracket to find its optimal weight. The optimization process employs a genetic algorithm, and the implementation utilizes **GeneticSharp**.

## Genetic Algorithm (GA)

What is a genetic algorithm?

A **genetic algorithm (GA)** is like a computer program that mimics how nature works. Imagine you have a group of creatures, and they all have different traits. Some traits help them survive better in their environment, while others don’t. The GA tries to find the best combination of these traits to solve a problem.

Here's how it works step by step as it relates to designing an arm bracket:

1. **Creating a Population**: Imagine you have a bunch of random creatures (here we call them "arm brackets"). Each arm bracket represents a possible solution to the problem. These solutions are like different recipes for solving the problem. The problem being solved is which chromosome weights less than the other chromosomes in the population.
2. **Natural Selection (Fitness)**: Just like in real life, some arm brackets are better suited for survival (they're the "fittest"). The GA looks at how well each arm bracket solves the problem (we call this the "fitness"). The fittest individuals get to "mate" and create new offspring. In the case of an arm bracket, the fitness is determined by the minimum weight in the population.
3. **Mixing Traits (Crossover)**: When two individuals "mate," their traits (or "genes") combine. It's like mixing arm brackets from two recipes to create a new one. This new arm bracket might be better than either of its parents.  
   A gene is either plate thickness, bolt size, bolt qty, stiffener thickness, or stiffener qty.
4. **Adding Some Variety (Mutation)**: Sometimes, the GA introduces small changes (mutations) to an arm bracket's traits. It's like adding a pinch of spice to a recipe. These mutations help explore new possibilities.
5. **Repeat**: The GA keeps doing this—evaluating fitness, mixing traits, and adding mutations—over many generations. Each new generation gets better at solving the problem.

A **genetic algorithm** is like a recipe optimizer that uses natural selection and mixing to find the best solution to a problem.

## GA Related to Structure Elements

Let’s relate to structural elements in the context of genetic algorithms (GAs) for structural design optimization.

1. **Chromosome Representation**:

* In a GA, a chromosome represents a potential solution (design) to the optimization problem.
* For structural design, each gene within the chromosome corresponds to a specific structural element (e.g., beam, column, joint, truss member).
* The entire chromosome encodes the complete design, including the arrangement, dimensions, and properties of these elements.

1. **Gene Encoding**:

* Genes can be encoded in various ways:
  + Binary Encoding: Each gene is represented by a sequence of binary bits (0s and 1s). For example, a binary chromosome might represent whether a particular beam exists (1) or not (0).
  + Integer Encoding: Each gene is an integer value representing a discrete choice (e.g., the number of beams or columns).
  + Real-Valued Encoding: Each gene is a real number representing a continuous parameter (e.g., length, thickness, material properties).

1. **Example**: Let’s consider a simple truss structure optimization problem:

* We want to design a truss with a fixed number of beams and joints.
* Each gene in the chromosome represents a beam (or a joint).
* Binary encoding: A gene value of 1 indicates the presence of a beam; 0 indicates its absence.
* The chromosome might look like: 1010010101, where each position corresponds to a beam or joint.

1. **Fitness Evaluation**:

* The fitness function evaluates how well a given chromosome (structural design) performs.
* For truss design, the fitness function considers:
  + Weight (minimize)
  + Stress (constraints)
  + Stability (constraints)
  + Other relevant factors

1. **Crossover and Mutation**:

* During evolution, crossover combines genes from two parent chromosomes to create offspring.
* Mutation introduces small changes to individual genes.
* These operations mimic genetic recombination and mutation in natural evolution.

1. **Iterative Improvement**:

* GAs iteratively evolve the population of designs.
* Each generation produces better designs by selecting, recombining, and mutating chromosomes.

## GeneticSharp

**GeneticSharp** is like a smart problem-solving tool inspired by how nature evolves species. Imagine it as a digital evolution simulator. Here's a simplified version:

1. **Chromosomes**: Think of these as your genetic recipe cards. You get one set from your mom and another from your dad. These cards contain instructions for building you, like eye color, height, and other traits.
2. **DNA Double Helix**: Picture a twisted ladder made of tiny building blocks. Each step of the ladder is a pair of nucleotides. These nucleotides spell out your genetic code.
3. **Genetic Algorithms (GAs)**: Now, GeneticSharp uses these ideas to solve problems. It starts with a population of "individuals" (like creatures). Each individual has its own set of "genes" (like those recipe cards).
4. **Natural Selection**: GeneticSharp evaluates how well each individual solves the problem. The best ones survive and pass their genes to the next generation. Over time, the population gets better at solving the problem.
5. **Mutation and Crossover**: Just like in nature, GeneticSharp mixes and tweaks genes. It combines the best traits from different individuals to create new ones. This helps find even better solutions.
6. **Optimization**: GeneticSharp keeps evolving the population until it finds the best solution to the problem. It's like a digital survival of the fittest!

**GeneticSharp** is a tool that mimics evolution to solve complex problems.

## The Design Process

**OptimizeArmBracket** follows the genetic algorithm procedure as discussed above.

A population is generated using a random selection of the following genes for each chromosome:

* **Bolt Diameter** – randomly select a bolt diameter starting at 0.625” to 1.50” incremented by 0.125”.
* **Bolt Qty** – randomly select an even number between 2 and 20.
* **Bracket Thickness** – randomly select an array index to select a thickness from the array (0.50, 0.625, 0.75, 0.875, 1.00, 1.25).
* **Stiffener Thickness** – randomly select an array index to select a thickness from the array (0.50, 0.625, 0.75, 0.875, 1.00, 1.25).
* **Stiffener qty** – randomly select either 0 or 2.

Each random selection of genes will make up a chromosome (solution). An analysis of the arm bracket is run using **ArmBracketLib**. The result of the analysis is used to provide the values for the chromosome. The **ArmBrackeLib** is run for each chromosome in the population.

When the population has been fully defined, **Natural Selection**, **Mutation and Crossover**, and finally **Optimization** (fitness) are used to create the next generation.

The fitness of the population is determined by the smallest weight. Since GeneticSharp uses the largest value to determine fitness, the fitness is equal to the arm bracket weight times -1.

Each population will contain from 75 to 200 chromosomes (solutions).

The algorithm will then iterate through many generations containing various mutations and crossovers until fitness stagnation occurs. Fitness Stagnation refers to a situation where the best chromosome’s fitness remains unchanged over several generations. When this occurs, it indicates that the genetic algorithm has reached a plateau, and further optimization is not occurring. For an arm bracket design, the fitness stagnation is defined as 10 iterations at which point the process is terminated.

At termination, **OptimizeArmBracket** will output the best chromosome (solution) which will contain the arm bracket with the optimum or lowest weight for the loads.

### ArmBracketLib

**ArmBracketLib** will analyze an arm bracket using Hock’s analysis procedure as defined in his 5/07/18 Excel workbook (Arm Bracket Design\_Version\_050718\_Latest.xlsm).