**Ideas:**

1. Add a new optimization algorithm, which should be based on backpropagation like Adam, to optimize parameters. (the problem here is if the algorithm adds new species to the system, the number of parameters is different from chromosome to chromosome, so it may not work. Maybe we can use a new population for each change in the number of parameters)
2. Use real number matrices, instead of binary representation
3. Use a pre-optimization phase to find the best number of components in the system and then use the best output and run the algorithm on it.
4. Three proposed structures:

* A pre-optimization step using GA to find the best set of components in the system to optimize the model(reduced simulation time to make this step more efficient, but the problem is, it may not show similarity to the full train time), after finding the best model structure, it will be used to optimize the model. (like in change-chr-ga paper, here we can use Adam algorithm to optimize parameters separately)
* A population will be initialized with the least number of components and with component mutation the system will try to find the best best set of components and optimize the system at the same time (like in Mousavi paper). The problem here is we may not be able to use Adam to optimize params separately.
* A population will be initialized with the least number of components and the algorithm tries to optimize the system based on till a certain amount of generation (for example till the best fitness changes no more) and then a new component/ more than one component will be add to the model and again the model will try to optimize the system till a certain amount of generation and …. It may be computationally more intensive than the other two models.

*(*[*https://doi.org/10.48550/arXiv.1703.01041*](https://doi.org/10.48550/arXiv.1703.01041)*)*

To automatically search for high-performing neural network architectures, we evolve a *population* of models. Each model—or *individual*—is a trained architecture. The model’s accuracy on a separate validation dataset is a measure of the individual’s quality or *fitness*. During each evolutionary step, a computer—a *worker*—chooses two individuals at random from this population and compares their fitnesses. The worst of the pair is immediately removed from the population—it is *killed*. The best of the pair is selected to be a *parent*, that is, to undergo *reproduction*. By this we mean that the worker creates a copy of the parent and modifies this copy by applying a *mutation*, as described below. We will refer to this modified copy as the *child*. After the worker creates the child, it trains this child, evaluates it on the validation set, and puts it back into the population. The child then becomes *alive*—*i.e*. free to act as a parent. Our scheme, therefore, uses repeated pairwise competitions of random individuals, which makes it an example of *tournament selection.*

*In each reproduction event, the worker picks a mutation at random from a predetermined set. The set contains the following mutations:*

*• ALTER-LEARNING-RATE (sampling details below).*

*• IDENTITY (effectively means “keep training”).*

*• RESET-WEIGHTS (sampled as in He et al. (2015), for*

*example).*

*• INSERT-CONVOLUTION (inserts a convolution at a random location in the “convolutional backbone”, as in Figure 1. The inserted convolution has 3 × 3 filters, strides*

*of 1 or 2 at random, number of channels same as input.*

*May apply batch-normalization and ReLU activation or*

*none at random).*

*• REMOVE-CONVOLUTION.*

*• ALTER-STRIDE (only powers of 2 are allowed).*

*• ALTER-NUMBER-OF-CHANNELS (of random conv.).*

*• FILTER-SIZE (horizontal or vertical at random, on random convolution, odd values only).*

*• INSERT-ONE-TO-ONE (inserts a one-to-one/identity*

*connection, analogous to insert-convolution mutation).*

*• ADD-SKIP (identity between random layers).*

*• REMOVE-SKIP (removes random skip).*

*A mutation that acts on a numerical parameter chooses the new value at random around the existing value. All sampling is from uniform distributions. For example, a mutation acting on a convolution with 10 output channels will result in a convolution having between 5 and 20 output channels (that is, half to twice the original value). All values within the range are possible.*

Add simulation steps, dt and other simulation parameters to the chromosome.

**Concepts:**

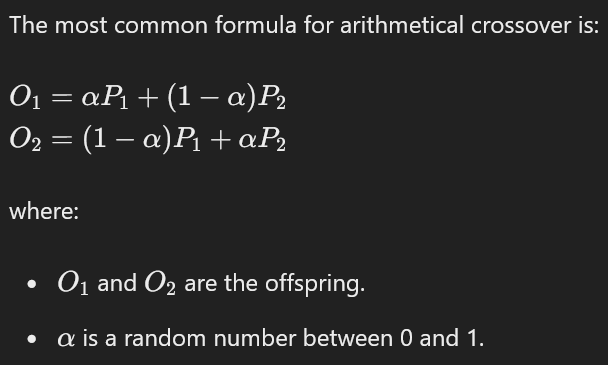
1. Genetic algorithm (GA)- based integer-valued optimization
2. Coefficient of determination (R2)
3. adaptive neuro-fuzzy inference system (ANFIS)

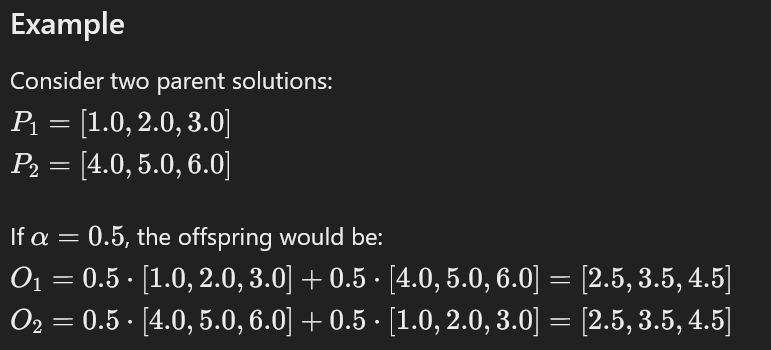
**Notes:**

a population size of 100 is set for the first generation and 50 for the followings.

Uniform operators of the algorithms are often used for this type of coding(continuous numerical problems).

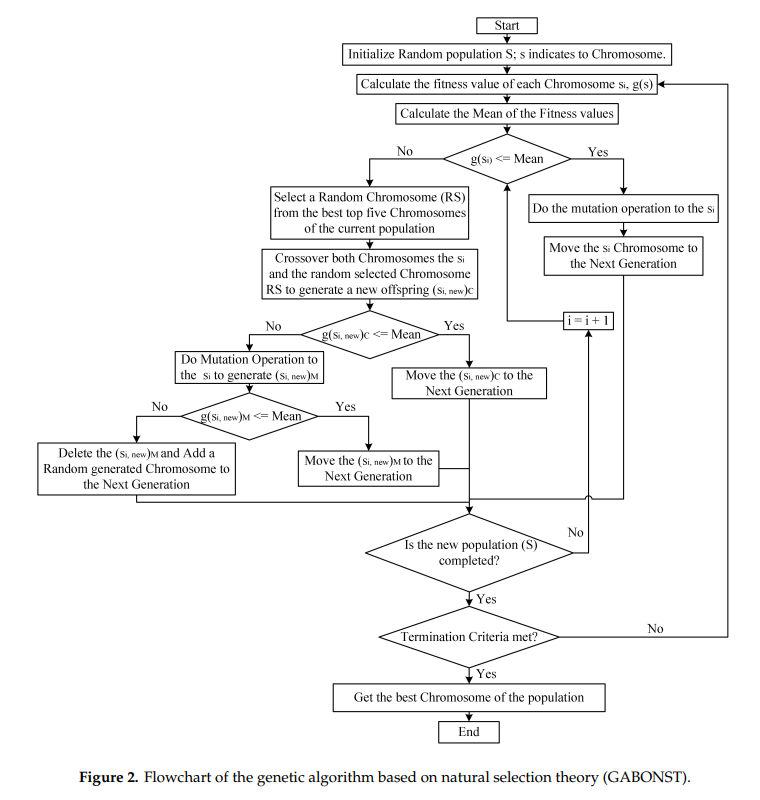
**arithmetical crossover**: is a type of crossover operator used to combine the genetic information of two parent solutions to generate new offspring solutions. This operator is particularly useful when dealing with real-valued representations of chromosomes.



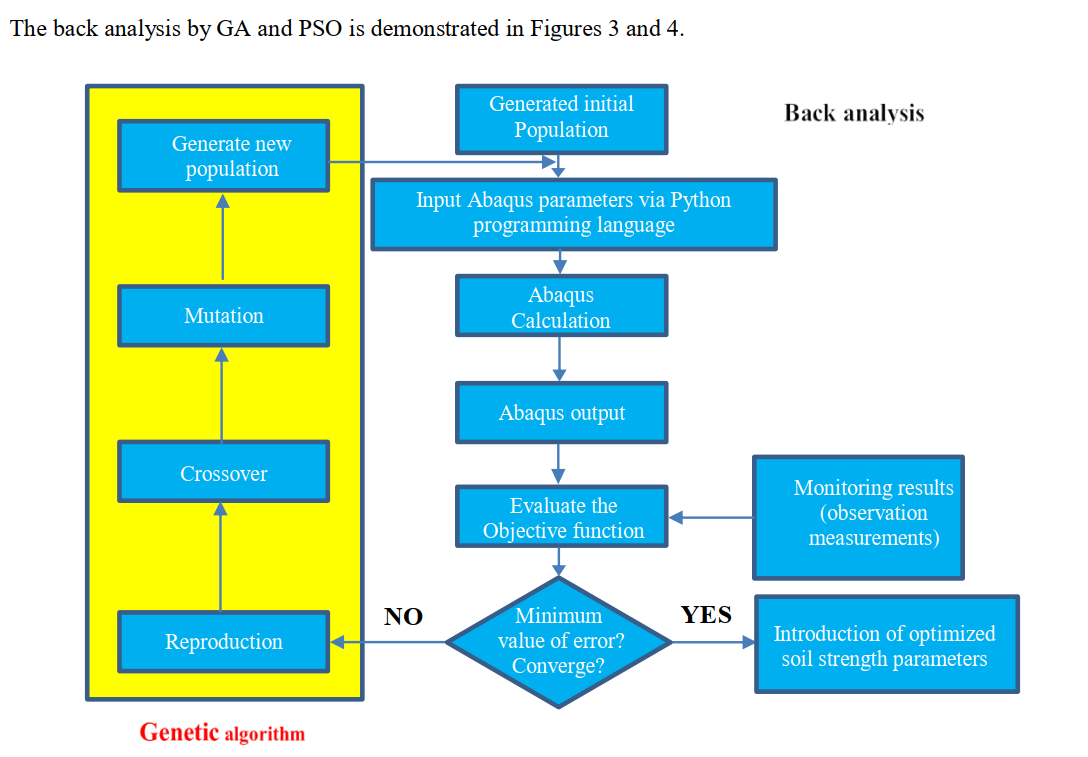


**NEAT** solves this problem by assigning a unique historical marking to every new piece of network structure that appears through a structural mutation. The **historical marking** is a number assigned to each gene corresponding to its order of appearance over the course of evolution. The numbers are inherited unchanged during crossover, and allow NEAT to perform crossover among diverse topologies without the need for expensive topological analysis

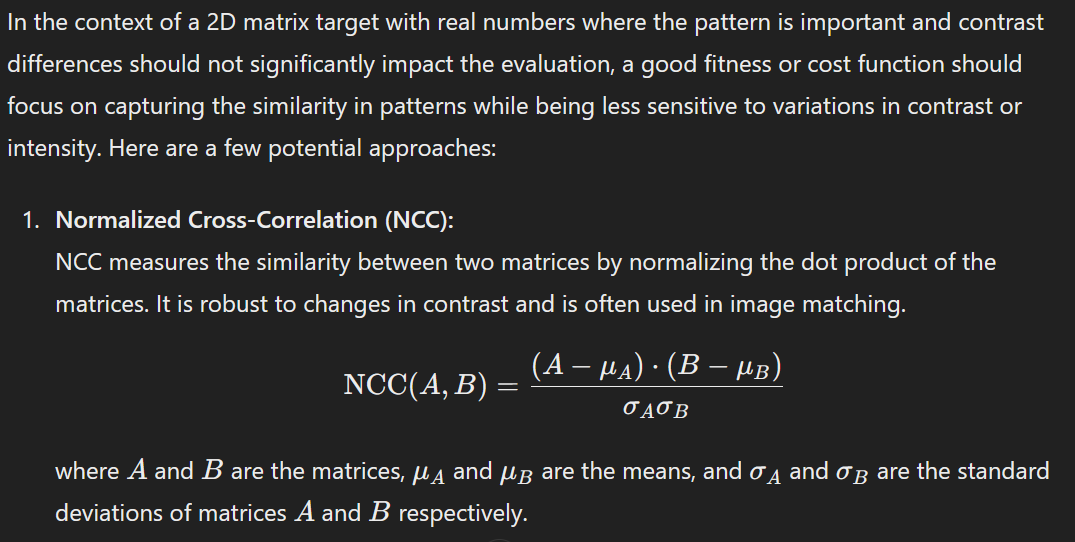
**GABONST algorithm:**



Connect different software together:



**Cost functions:**

****

