# Report

## Optimizer for the evolutionary generated frame structures


Authors:

Joanna Baczewska -UI, visualization

Pawe? Bielecki - Implementation of classes of a bridge, physics simulator, math utilities for calculating vectors in two dimensions.

S?awomir Klimowski - Neural Network and interface functions for AI


### Description:

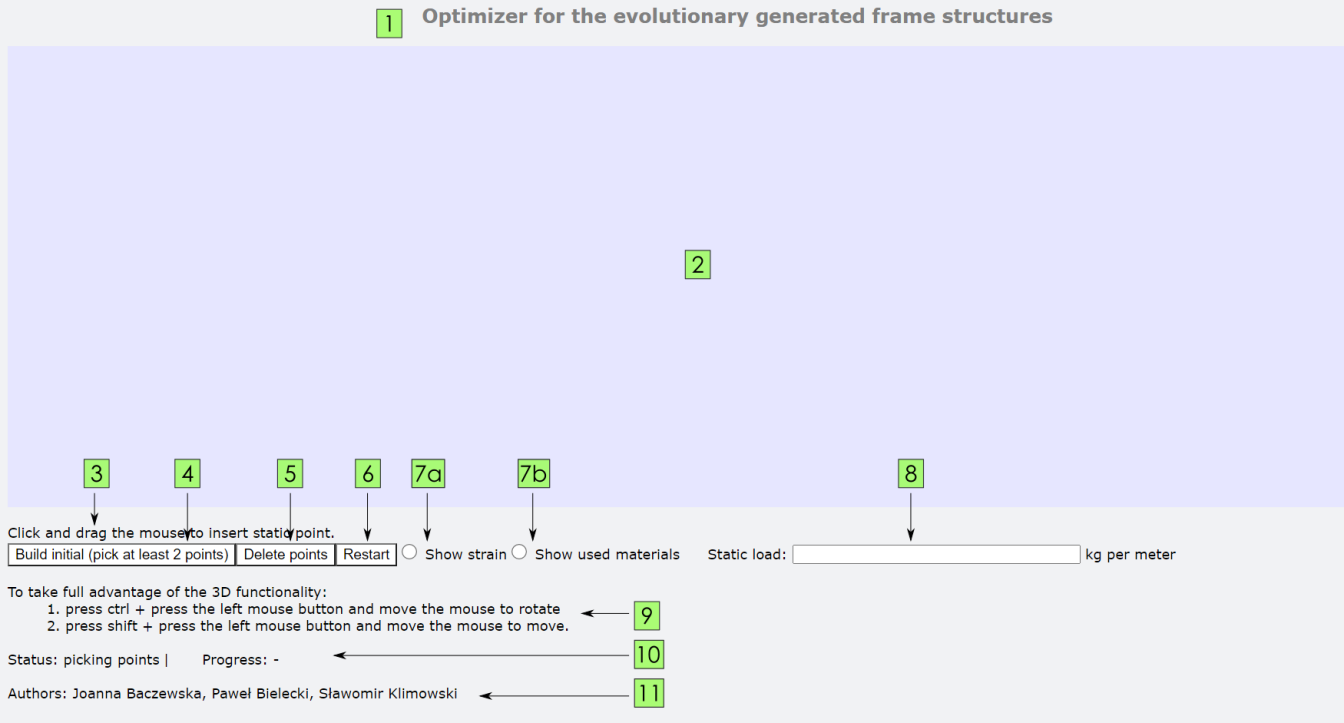
**Project is concentrated around usage of the Artificial Neural Networks as a problem solving and optimizing algorithms. It consists of standalone modules which divide code into functional parts.**

### Modules:

1. User Interface ? preparation of the problem to solve by program and creation of a graphical representation of the solution for given problem with animations.
2. Physics engine - handling of all required calculations and simulation processes of the program.
3. AI architecture - model of neural network and its various interface functions necessary for the process of building and altering the structure.?

### GUI

Starting the application automatically opens a new page in your default browser. The appearance of GUI is as follows:




1. Project title
2. Canvas - a scene on which a visualization of a bridge simulation is displayed
3. Text field that displays a tooltip depending on the stage of using the project

   Menu
4. button that starts the bridge simulation if at least two points are selected in canvas
5. button to delete the entered points before starting the simulation, enables to re-enter the points
6. button to stop the simulation and clear the scene so that you can enter points and turn on the simulation again.
7. Radio buttons to select the way of displaying the bridge:

    a. showing the loads in accordance with the color characteristics green - minimum, magenta - maximum
    b. showing the bridge according to the colors of the materials used [steel / asphalt / wood]
8. input field to enter an additional static load for which the bridge is to be generated [kg per meter]

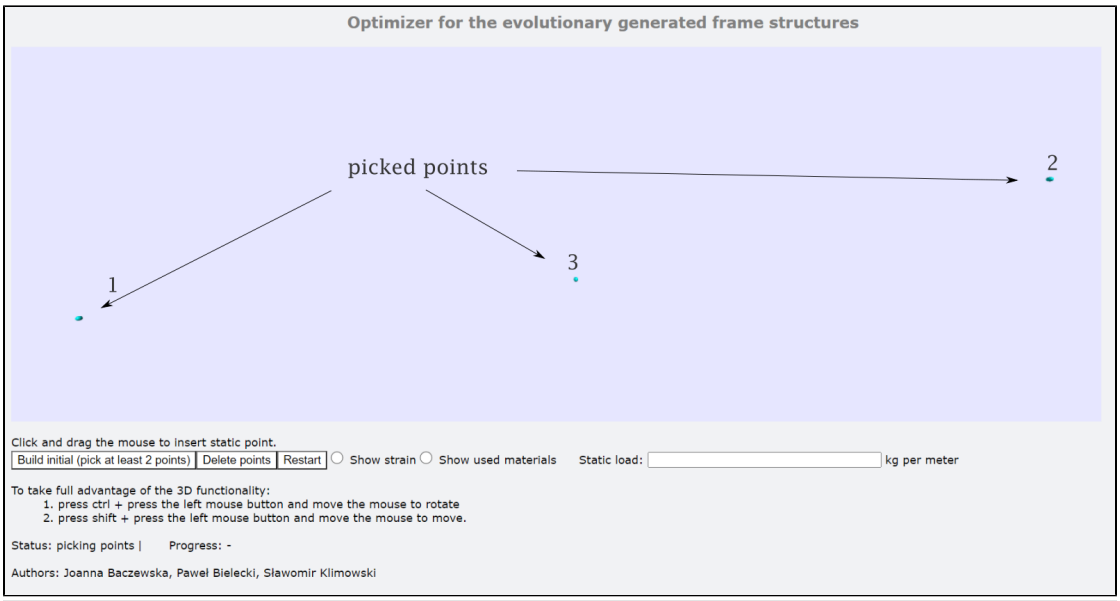
Information fields

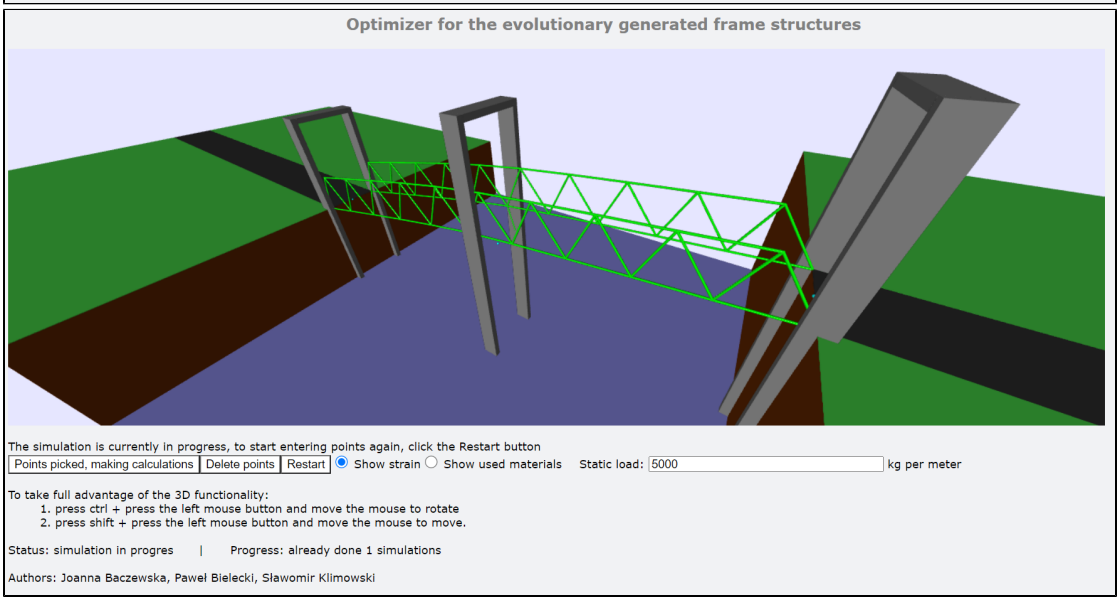
9. Help instruction for using the camera rotation and moving in the 3D scene
10. Text fields informing about the status. Possibilities depending on status:
    a. Status: picking points
    b. Status: simulation in progress
    c. Status: simulation ended, bridge broken
    d. Status: bridge simulation completed successfully, bridge stable
    The progress field displays the number of simulations that have started so far from the beginning after entering the points
11. Information about the authors of the project?

## Instruction and preview

1. After starting the application, select the points in the following order:
   1 - the beginning of the road
   2 - the end of the road
   -? then more optional static points can be added. They can be used in the generation process if it will be easier to obtain an optimized bridge.
   To select a point, press down left mouse button in the stage area, the selected point will be set in the place where the mouse button will be released. Before releasing the mouse button, it is possible to change the position of the point by moving the mouse. In case of a mistake you can press "Delete points" and choose the points again.
   In addition you can enter Static load.
   If you picked at least 2 points and want to start generating the bridge, press the "Build initial" button

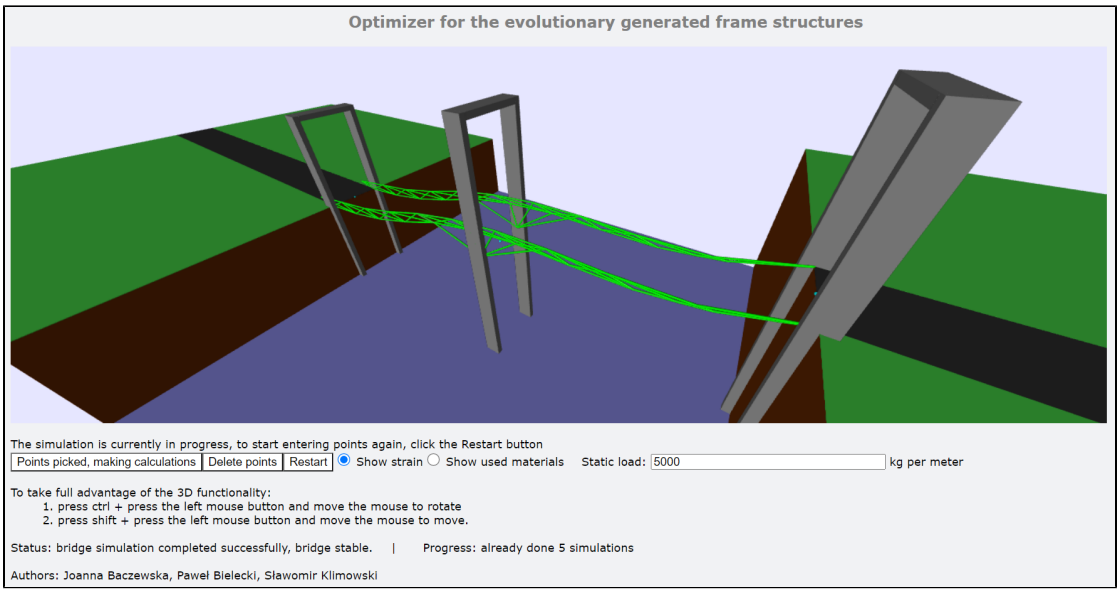
2. After starting the simulation, the first version of the bridge is visible first, each time very similar to the one shown in the picture number 2. The camera change position and terrain is displayed (generated in a position consistent with the entered points). If additional static points are used, additional supports are also displayed at the points that are connected to the bridge in a given simulation. During the simulation, you can freely change the way the bridge is displayed (the "Show strain" way is present in the preview) and modify the Static load. If you want to stop the simulation, you can click "restart". If not, we wait for the end of the simulation, which is currently "in progress" - it is indicated by the status displayed at the bottom of the page.

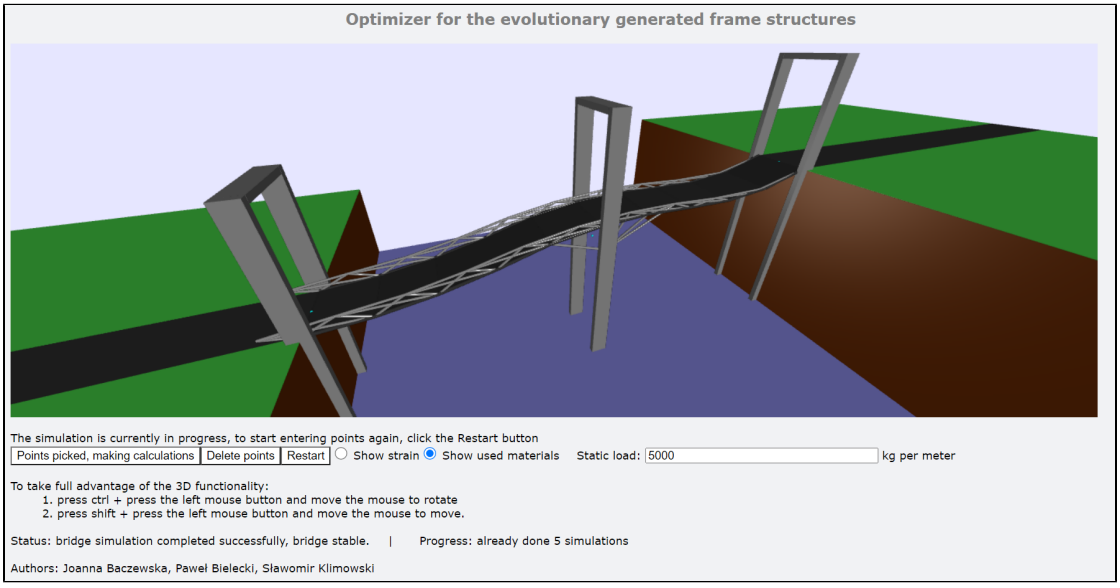



3. The bridge changes and shows the next generated steps. The number of simulations performed so far is written after "Progress". After the simulation is finished and the result is obtained (stable bridge or broken bridge), the "Status" field changes according to the result.

4. During the simulation but also after the calculations are completed, you can change the way the bridge is shown (for example in the picture 4 you can see the "show used materials" variant). You can also rotate the camera (the axis of rotation is placed in the middle of the bridge by default) and you can move it (as the instruction in the UI informs). An example of changing the position of the camera is shown in the picture. To simulate another bridge, you can click 'Restart'.




## Physics engine

The bridge is stored as two list: list of connections and list of points. Every connection has references to two joints. Furthermore it has information about its material, mass, strength and force rates. Every joint has information about inertia moments of all connections which are connected to it. Furthermore it stores information about force vector, velocity vector and position. A joint could be stationary or not. Being stationary mean that the joint cannot be moved during the simulation.

The main function of the physics engine is *simulateTimeStep* which makes one step in simulation. It uses auto-adjustment time step. When the simulation accuracy is enough the time step is increased. Otherwise it is decreased and the last step is cancelled. The error is calculated by calculating one step with a length 2?t and two steps with a length ?t. Relative differences of forces vectors are summed and the sum is the error:

$$\Delta = \sum_{i=0}^{n-1} \frac{\left\| \vec{F}_{i,1} - \vec{F}_{i,2} \right\|}{\left\| \vec{F}_{i,1} \right\| + \left\| \vec{F}_{i,2} \right\|}$$

For movement simulation it uses basic physics equations:

$$\frac{\vec{dv}}{dt} = \frac{\vec{F}}{m} \Rightarrow \Delta \vec{v} = \frac{\vec{F} \cdot \Delta t}{m}$$

$$\frac{\vec{dr}}{dt} = \vec{v} + \frac{\vec{dv}}{2\,dt} \Rightarrow \Delta \vec{r} = \left( \vec{v} + \frac{\Delta v}{2} \right) \cdot \Delta t$$

On the left side there are physics formulas, on the right side there are approximated formulas for time steps.
Forces are calculated from a gravity and mass of the connections and from spring forces which are produced by connections:

$$\vec{F} = \frac{-\vec{v}}{\|\vec{v}\|} \cdot k \cdot \left( l_0 - l_c \right) + g \cdot m$$

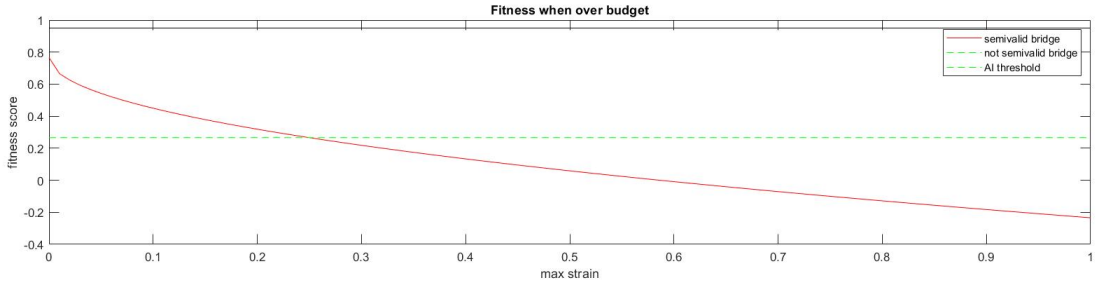
where *v* is vector of position (jointA.position - jointB.position or reversely - depend for which joint forces are calculated), $l_0$ is original length of the connection, $l_c$ is current length, *k* is compression force rate or stretch force rate - it depends if current length is greater or lesser than the original length.

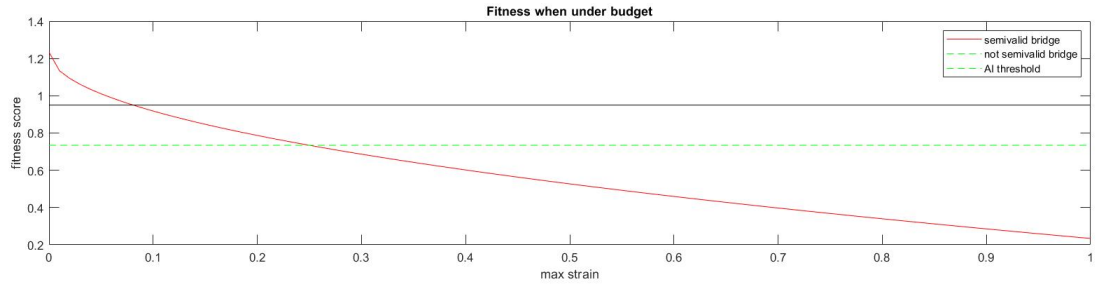
For executing all process of simulation there is a *simulate* function which checks if the bridge survives and it collects useful information about the bridge's straightness. It uses *simulateTimeStep* in a loop.
It optimizes parameters of simulation to minimize the potential energy of the bridge as fast as it is possible. The simulation is not real - the bridge's velocity is contained to keep optimal size of time step.

## AI

Artificial Intelligence is based on NEAT model (Neuro-Evolution of Augmenting Topologies) which unlike most well known machine learning algorithms does not require predefined output of the network. This model classifies the correctness of the model based of an arbitrary value called fitness that can be calculated indirectly to the outputted values. For our purposes fitness is calculated following the recipe implemented in score function in ai module from tbneuralnetwork package.

The chart below represents values corresponding to the maximum value of strain recorded during simulation in two edge cases (when bridge cost is over budget and when it is below):



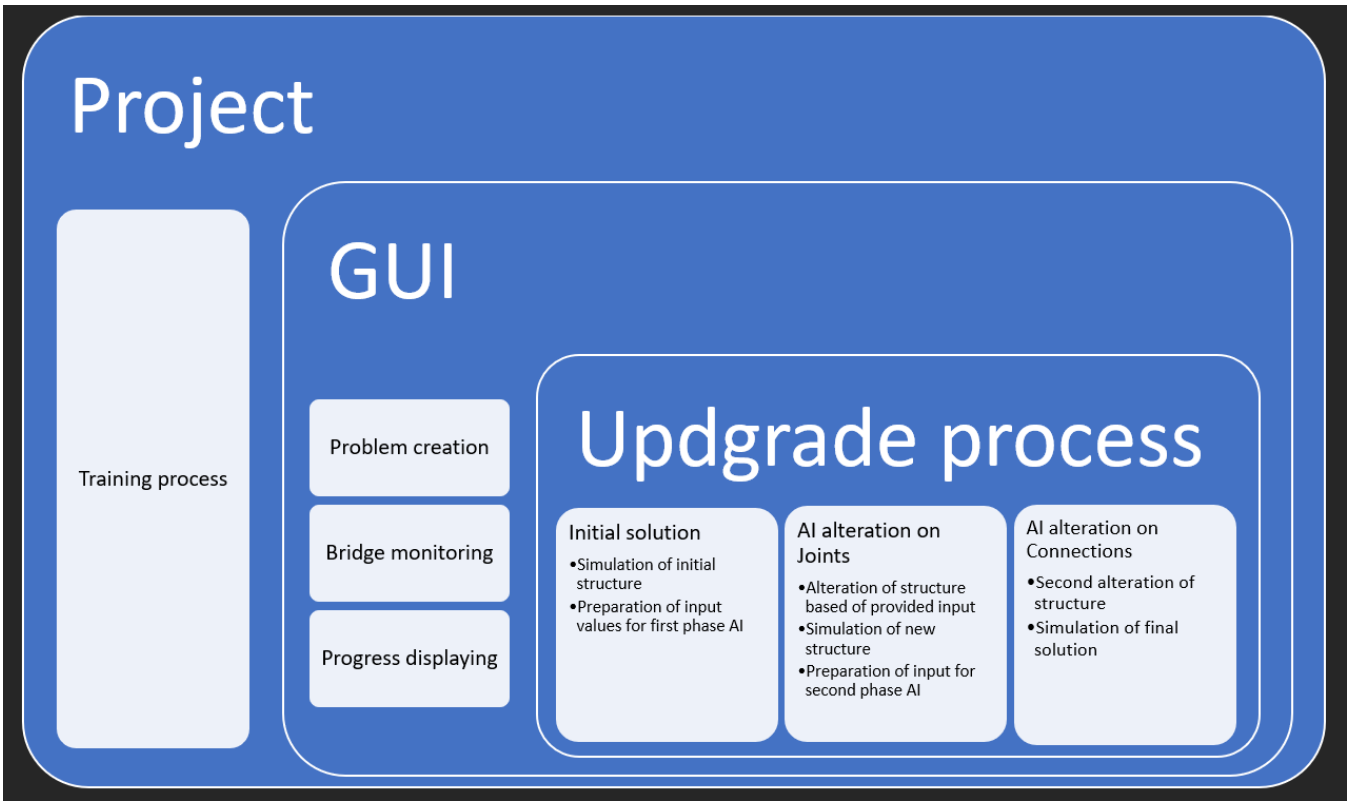

AI is separated into two phases, first that alters bridge structure based on joints using these interface functions from module neuralnetworkfunctions from tbneuralnetwork package:

- addJoint()
- removeJoint()
- moveJoint()
- addConnection()

The second phase modifies connections of the model using functions:

- removeConnection()
- changeConnectionMaterial()

## Block diagram of the project:




## Code documentation:

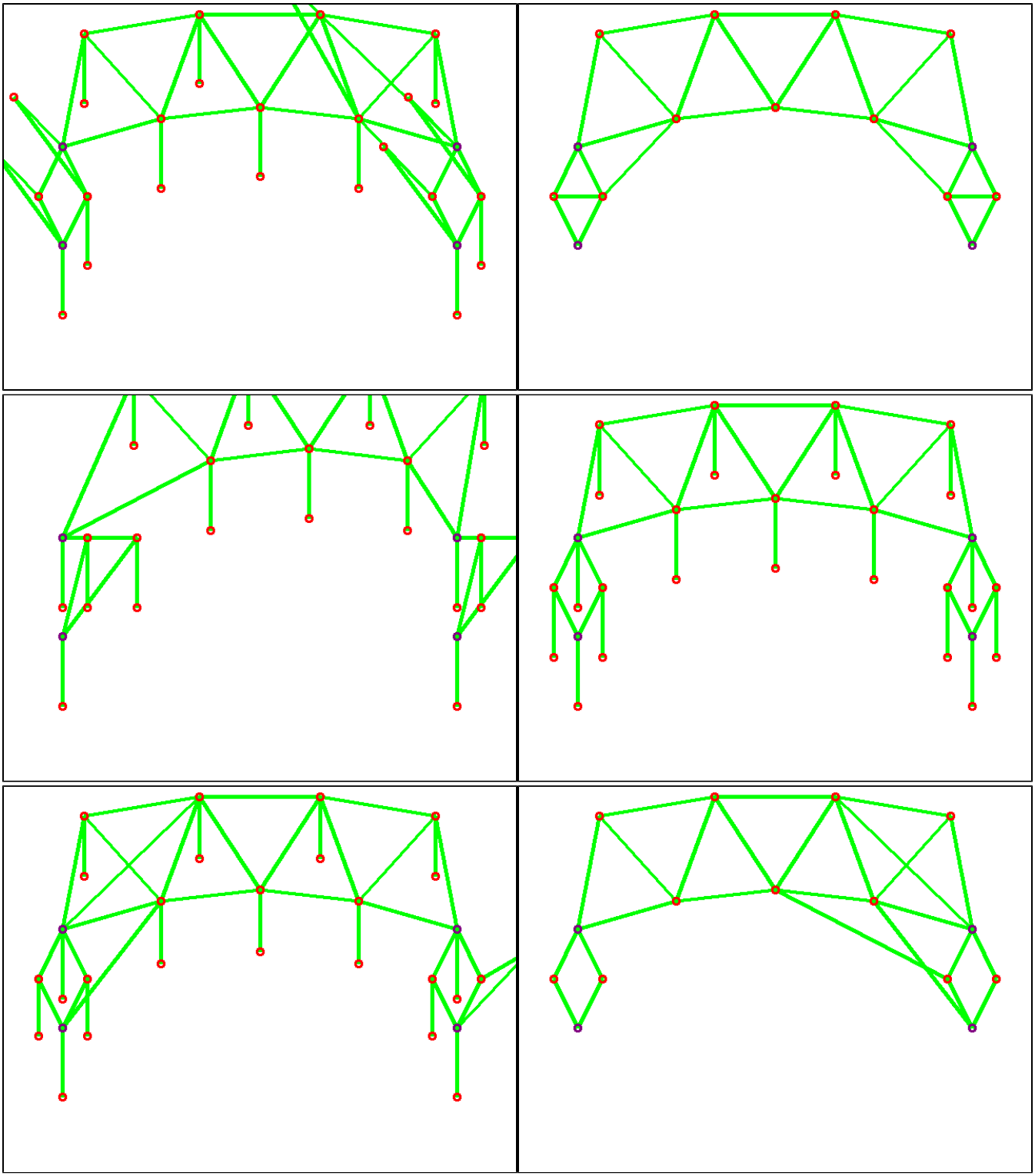
Code has been documented using the standard Python documentation technics and it can be easily accessed by built in methods ( help, dir) and pydoc program.

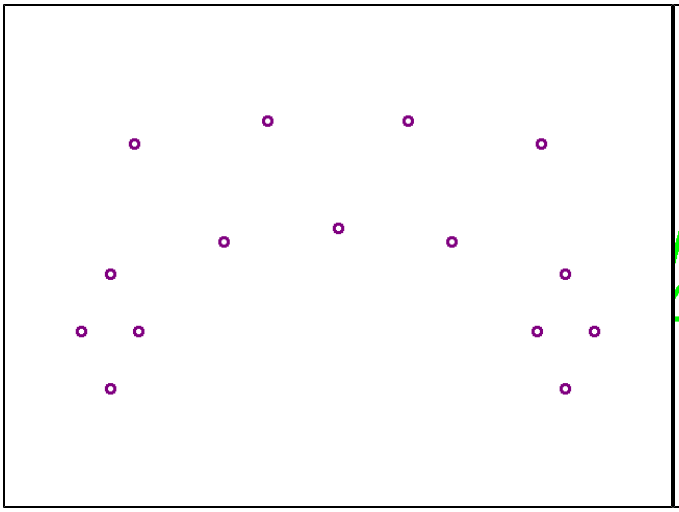
## Training process:

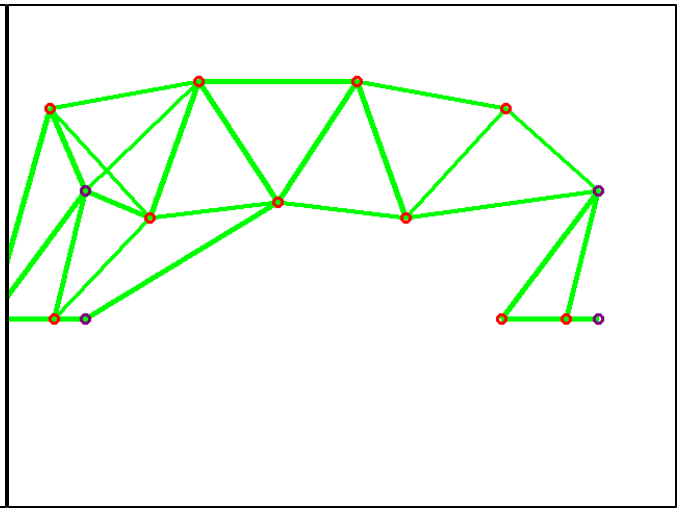
The training process has been drastically shortened due to time constraints and resulting model should be used only as proof of concept. The biggest obstacle in preparing a fully trained model is necessity to perform whole physics simulation for every variant of the bridge structure. Over time the time needed per simulation was significantly shortened but every improvement meant that AI must have been retrained in new environment. In the final version whole process of training for one bridge structure took around 6 hours in best case scenario, although some structures proved themselves more difficult to simulate and could take over a day to finish. Despite that we repeatedly recorded improvement of the structure within first generation.

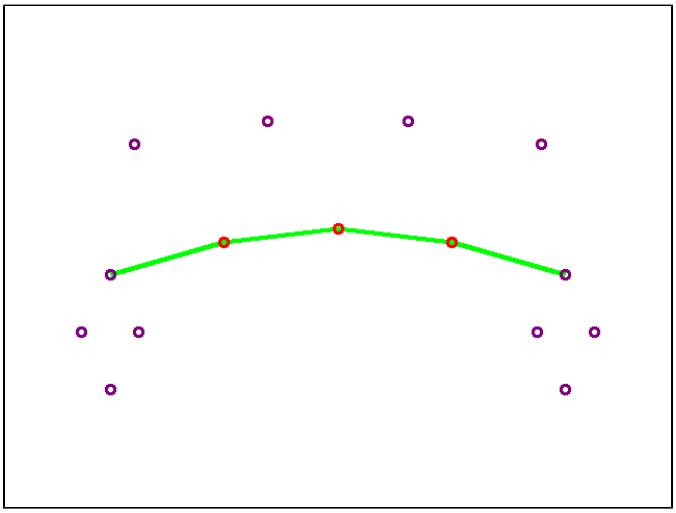
Training process is mostly linear and every genome in generation is independent to others which makes ground for multi-threading. Everything starts with first simulation of the provided structure, then the result is translated into input vectors for first phase of AI which uses set of functions that alter model of the bridges based on it joints. On every one of the possible structures another simulation is performed to evaluate the fitness of solution. This process is on default set to go up to user specified number of generation with 50 genomes per generation. After every generation new one is prepared unless one of genomes achieved desired results then first phase is considered finished and new input vectors are prepared based on the proposed structure. Usually criteria is met within first generation. Second phase similarly to the first creates new solutions to be evaluated but this time alterations are made on connections itself.

?Below there are few examples of proposed solutions :









Green lines represent any type of connection and small circles are markers for positions of the joint, where red color means that this joint is not stationary and purple ones are, although if non stationary point is not connected with any other joint then it is treated as stationary point.

The last example was one of the most difficult ones to simulate taking over one whole day to finish where most of the cases takes no longer than minute.? It is the sole reason why in main program we create relatively complex structure as starting point. Well trained AI could possibly create whole structure from scratch but necessary time needed to perform all of the calculation would greatly exceed amount we had left to deadline.?

## Summary:

We managed to finish all assumed goals in project roadmap on time, thou we can see room for improvement in few aspects. Working on this project showed us how important is well organized teamwork. Despite that most of the code was written as standalone modules we encounter many problems while linking them together. This problem showed mostly while we were optimizing physics engine, where every minor change in functionality meant not only retraining of the AI but sometimes altering the structure of the network or changes into evaluation process.
In the end this project become great proof of concept with promising result. ?