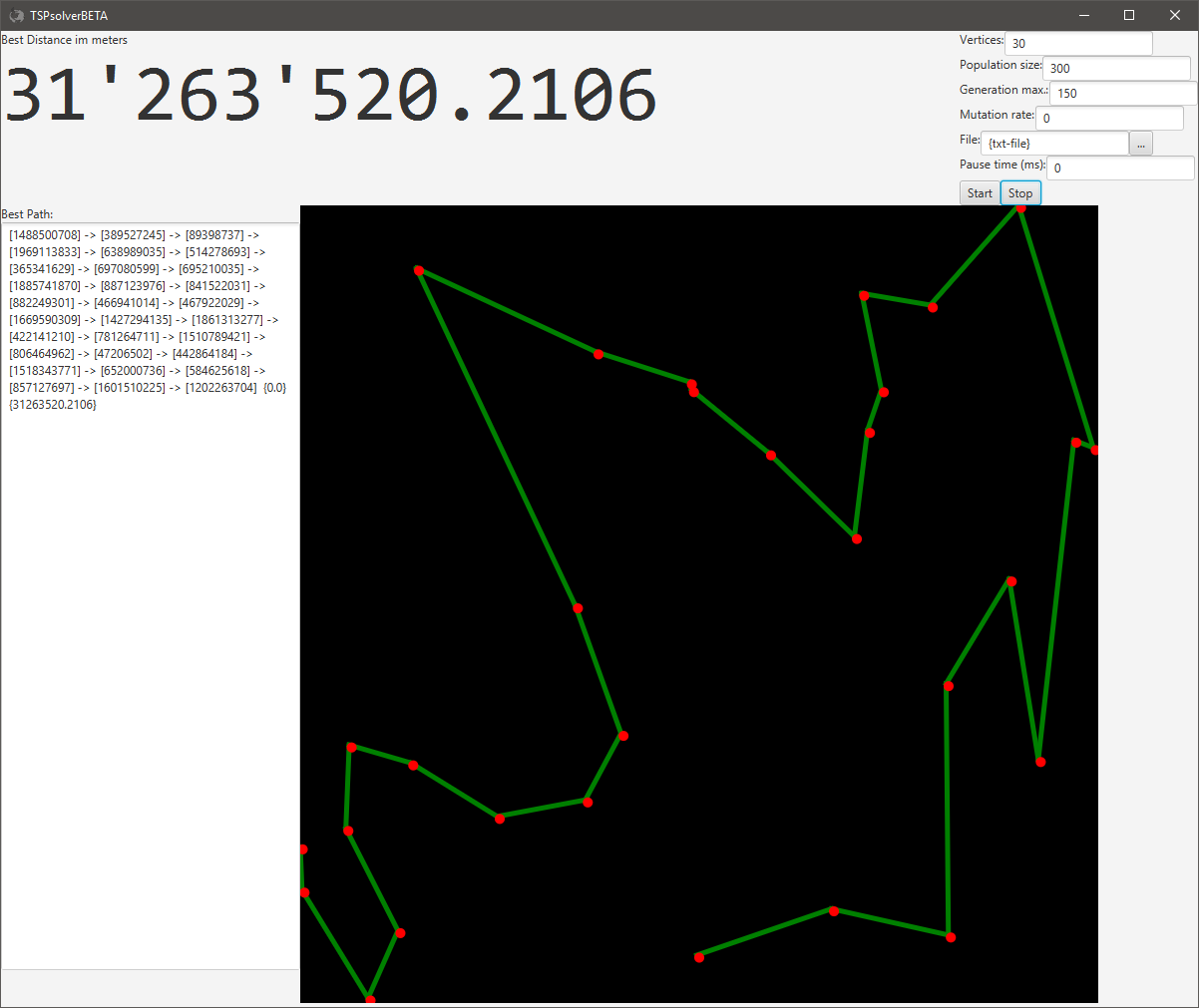
TSPsolver

# Basic Idea

This project is about solving the famous Traveling Salesman Problem with a Genetic Algorithm. This document provides instruction how to handle the application as well as how to interpret its findings. Detail information about the problem as well as the applied algorithm can be found in the reader.

# GUI

The picture below shows the graphical user interface of the application. Please be aware that the requirement of calculating a closed path was omitted in this project (Hamiltonian path instead of Hamiltonian circle).



# Parameters

Parameterization is a crucial part when applying a genetic algorithm. The following constants can be set by the user.

## Vertices

When no file is indicated, the application will create random vertices. The number of vertices being drawn can be set in the first parameter. All positive integers are valid.

## Population size

Specifies the number of elements being held in one population. All positive integers are valid.

## Generation max.

Specifies the number of evolutions being performed with a single population. After the end is reached, the application instantiates a new population and starts over again.

## Mutation rate

Specifies how much percent of an element may be affected by random exchange interventions of the element’s concrete solution path. All positive integers are valid. Nonetheless, it is advisable to choose a value between 0.0 and 1.0. A value of 3.0, for instance, would engender that 300% of the solution path is randomly changed.

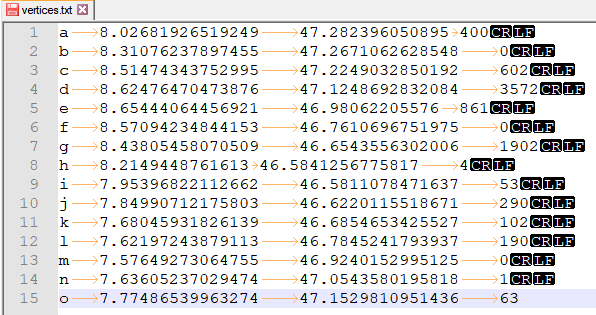
## File

In order to apply the algorithm to a concrete problem, the user may provide a text file with coordinates. In doing so, the number of vertices specified in the first parameter gets disregarded, provided the file path is valid. Additionally, the imported file has to confirm the following line structure.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| required | | | | | optional | |
| vertex name | tab | latitude | tab | longitude | tab | height |

The coordinates must not necessarily stay in the World Geodetic System (WGS) form as the method drawPaths will paint all vertices proportionally onto the black area.

Each line represents one vertex. The lines can be accumulated by means of a carriage return and a line feed (CRLF). An example of a valid text file is shown below.



## Pause time (ms)

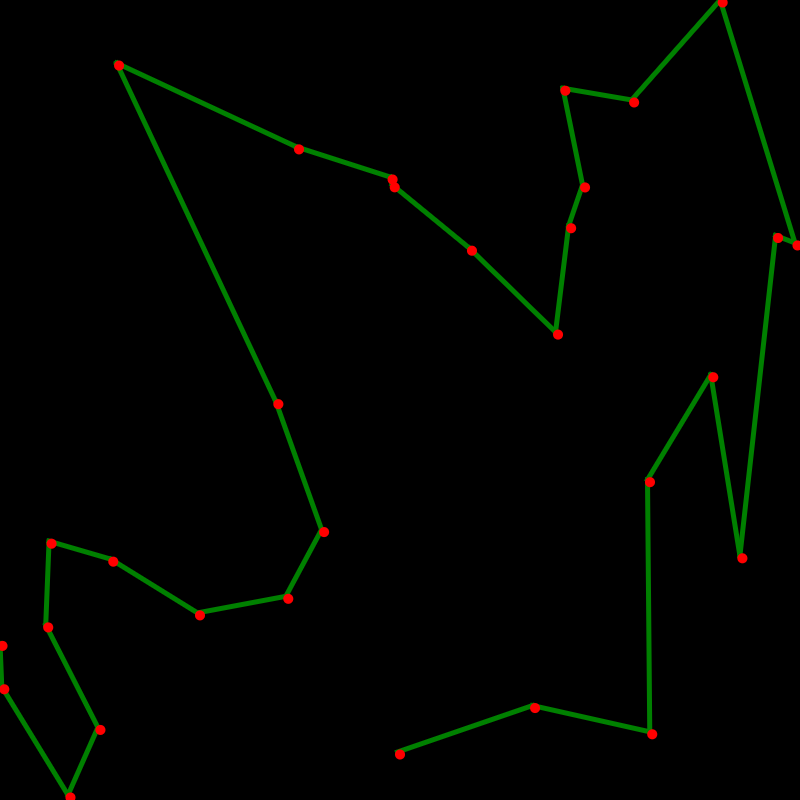
The pause time represents the number of milliseconds the application is forced to wait after each evolutionary iteration. This enables to better observe the numerous improvement steps. All positive integers are valid.

# Results

The current best path (= minimal path) is shown in the following ways.

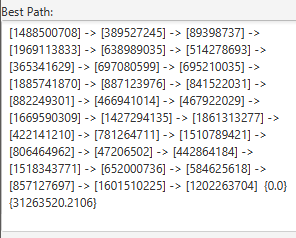
## Graph

Always the best solution is being shown in the black area. Once the algorithm finds a better solution, the painting gets redrawn.



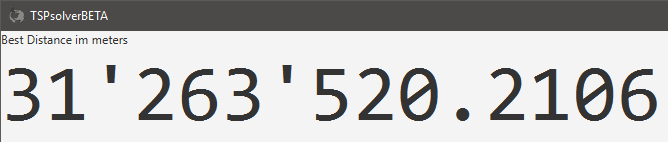
## Path

The paths shown in the black area is specified in the text area left to it. When working with random vertices, the identity hash code of the vertex object serves as vertex name. At the end of the path, the total distance is shown as a double value.



## Distance

The total distance of the current best path is shown in the upper left corner. The distance gets calculated by means of the VertexSet’s object function getTotalDistanceWGS.



# PopBook

Detailed information about each generation of a population is logged in a text file stored at [project path]/PopBook. It enables to explore the evolution procedure.

