

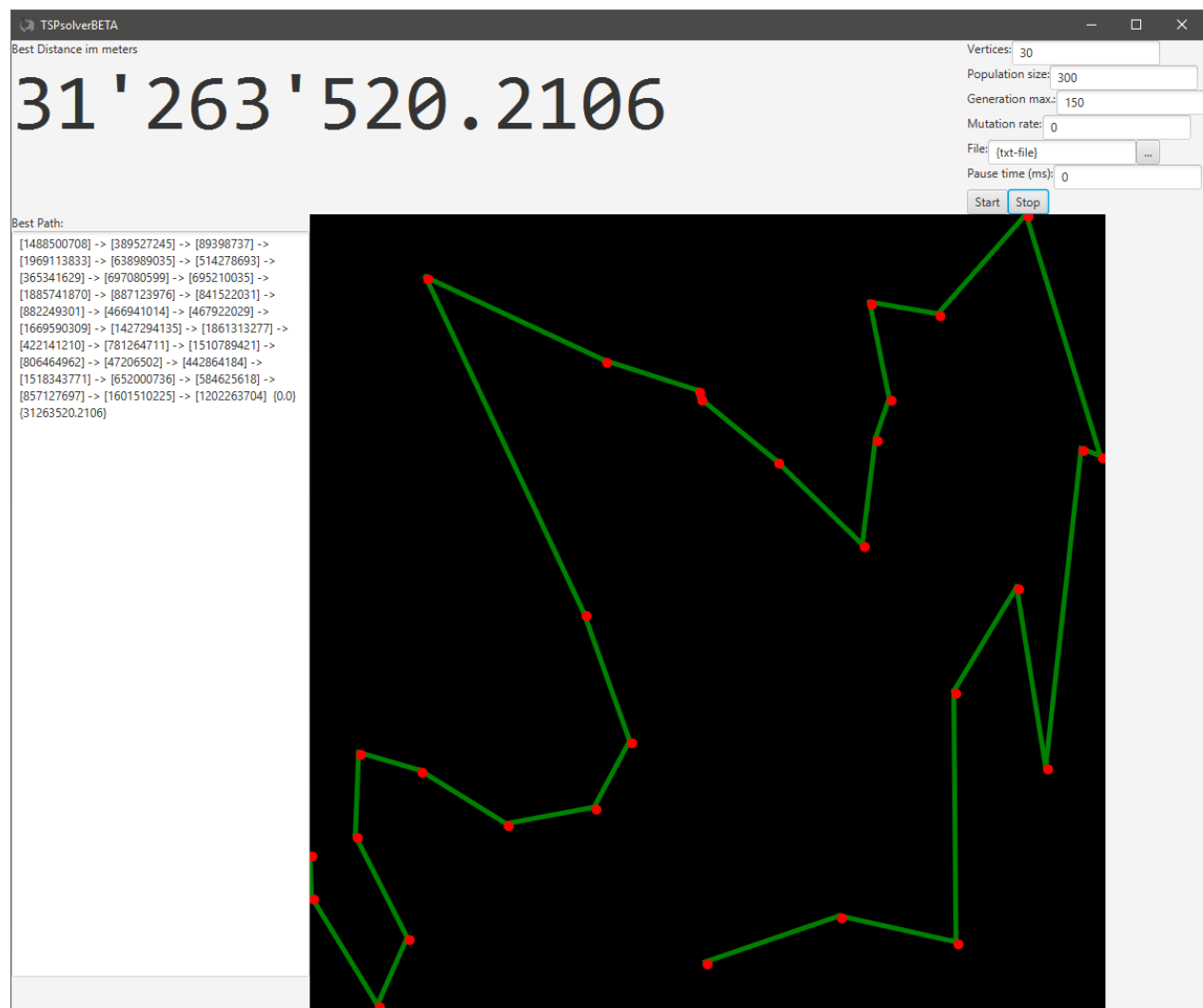
TSPsolver

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.

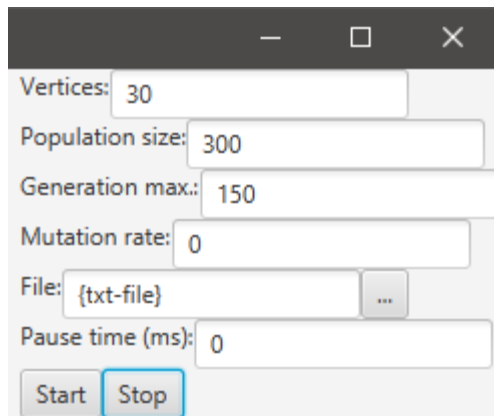
GUI

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.

A screenshot of a software window with a dark title bar containing standard window controls (minimize, maximize, close). The window contains several input fields and two buttons. The fields are labeled 'Vertices:', 'Population size:', 'Generation max.:', 'Mutation rate:', 'File:', and 'Pause time (ms):'. The values entered are 30, 300, 150, 0, '{txt-file}', and 0 respectively. The 'File:' field has a small button with three dots to its right. At the bottom are 'Start' and 'Stop' buttons.

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.

```
vertices.txt
1 a → 8.02681926519249 → 47.282396050895 → 400 CRLF
2 b → 8.31076237897455 → 47.2671062628548 → 0 CRLF
3 c → 8.51474343752995 → 47.2249032850192 → 602 CRLF
4 d → 8.62476470473876 → 47.1248692832084 → 3572 CRLF
5 e → 8.65444064456921 → 46.98062205576 → 861 CRLF
6 f → 8.57094234844153 → 46.7610696751975 → 0 CRLF
7 g → 8.43805458070509 → 46.6543556302006 → 1902 CRLF
8 h → 8.2149448761613 → 46.5841256775817 → 4 CRLF
9 i → 7.95396822112662 → 46.5811078471637 → 53 CRLF
10 j → 7.84990712175803 → 46.6220115518671 → 290 CRLF
11 k → 7.68045931826139 → 46.6854653425527 → 102 CRLF
12 l → 7.62197243879113 → 46.7845241793937 → 190 CRLF
13 m → 7.57649273064755 → 46.9240152995125 → 0 CRLF
14 n → 7.63605237029474 → 47.0543580195818 → 1 CRLF
15 o → 7.77486539963274 → 47.1529810951436 → 63
```

Pause time (ms)

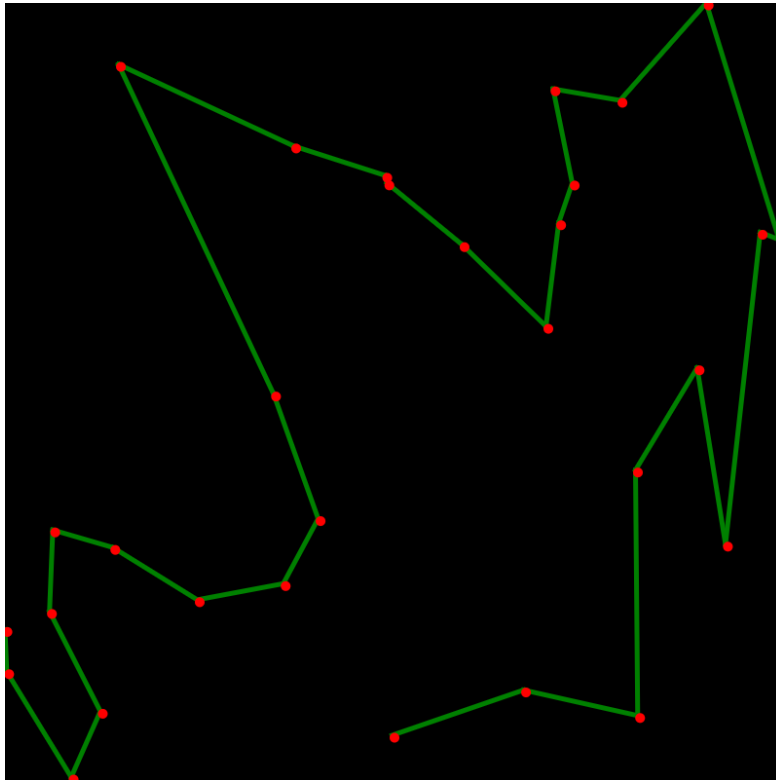
The pause time represents the number of milliseconds the application is forced to wait after each evolutionary iteration. This enables to better observe the numerous improvement steps. All positive integers are valid.

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.

Best Path:

```
[1488500708] -> [389527245] -> [89398737] ->
[1969113833] -> [638989035] -> [514278693] ->
[365341629] -> [697080599] -> [695210035] ->
[1885741870] -> [887123976] -> [841522031] ->
[882249301] -> [466941014] -> [467922029] ->
[1669590309] -> [1427294135] -> [1861313277] ->
[422141210] -> [781264711] -> [1510789421] ->
[806464962] -> [47206502] -> [442864184] ->
[1518343771] -> [652000736] -> [584625618] ->
[857127697] -> [1601510225] -> [1202263704] {0.0}
{31263520.2106}
```

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.

