# Directory Structure

The three main directories contain:

* Distributable: The currently compiled code for the double constrained spatial interaction model
* Example Data: A set of idealised small system data to run through the model so you can make sure that any changed you make are behaving as you would like them to
* Source Code: the full source code for the model and all external libraries developed by myself and the Leeds MASS group

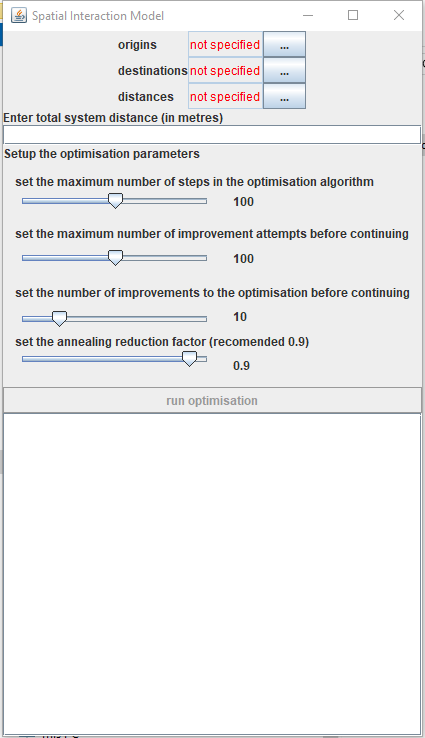
# Adapting Beta Parameter(s)

* Currently the Beta parameter is tightened right down so that it acts across the whole model. I would recommend changing this so that you have a beta parameter for each ‘type’ of individual in the model rather than looking at a parameter for each area, this will become unwieldly quickly and likely to result in an overly fit model.
* The beta parameter is represented by an array of doubles currently the array is one dimension containing one element. To change this the places that would need to be adapted are:
  + *Run* method in *SpatialInteractionModel.java*: this is where the beta parameter array is initially dimensioned and setup.
  + *Calculate* method in *DoubleModel.java*: where the flow is calculated you would need to define how to access the correct beta parameter from the array depending on the classification of the origin or destination you are looking at.
  + *CalculateAi* method in *DoubleModel.java*: where the equation to work out the cumulative denominator is calculated you would need to define how to access the correct beta parameter from the array depending on the classification of the origin or destination you are looking at.
  + *CalculateBj* method in *DoubleModel.java*: where the equation to work out the cumulative denominator is calculated you would need to define how to access the correct beta parameter from the array depending on the classification of the origin or destination you are looking at.
  + *suggestChange* method in *DoubleModel.java*: the beta index would need to be adjusted back to pick a random index from the length of the array, the code to do this is commented out above where the betaIndex = 0; line sets the index to always be 0.

# Running The Model

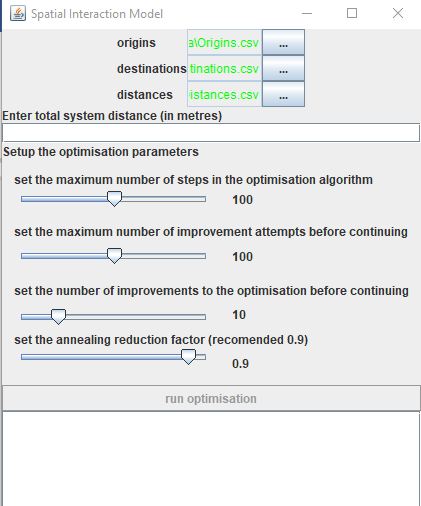
Run the OS\_SIModel.jar file from the distributable folder on any computer running Java 8 or above (note the lib folder must be kept with the jar file is copied to different machines as this contains packaged dependencies required by the model code.

When OS\_SIModel.jar has been launched you will see the screen below

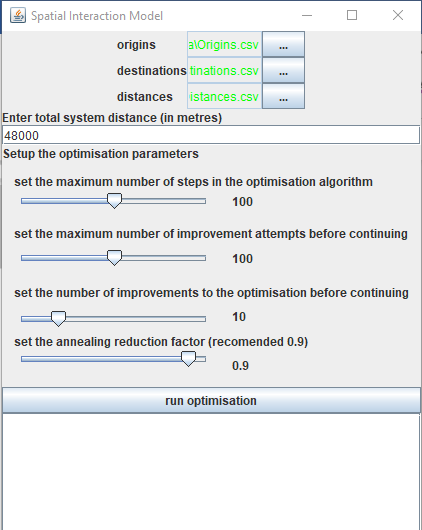


First load the origin, destination and distance files by clicking on the button next to the text box marked with ‘…’. A file chooser will be launched for each one, navigate to the folder containing the file you would like to load, select the files and click ‘Open’ or double click the file itself.

When all three files have been successfully loaded (it may take a few minutes to load the distance file if this is several million records in length) you will see the path to the files written in green in each of the text boxes as below.



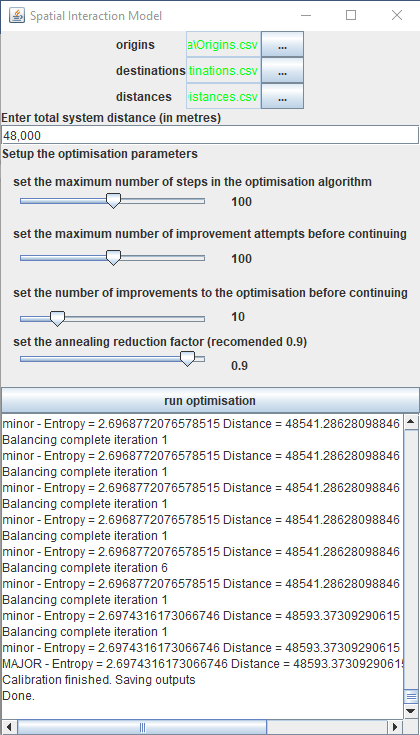
Next you need to enter the overall distance you would like to constrain to in the distances text box below the three file load boxes. Below it shows 48000 has been entered into the system distance input box, when this has been entered correctly the ‘run optimisation’ button will become enabled.



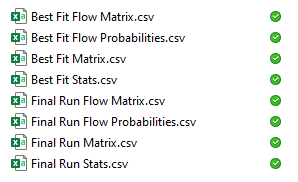
The sliders below the system distance input box control how hard the optimisation algorithm will work. The number of **steps** represents the maximum number of iterations the algorithm will perform. The maximum number of **improvement attempts** is the maximum number of times the algorithm will try and step forwards in any **step** iteration. The number of **improvements** that will be made before continuing controls how many improving changes are be made before the algorithm will move to the next **step.** A combination of these two settings controls when the algorithm will move to the next **step**. In the screen shot above if the algorithm hits 10 **improvements** or 100 **improvement attempts** within one **step** it will move to the next **step**. The **annealing reduction factor** is the rate at which the algorithm will ‘cool’ or stop accepting non-improving changes, 0.9 is generally a good rate.

Note: the number of **improvement attempts** and **improvements** are multiplied by the sample size for the problem at hand. The sample size for this model is the parameter attempting to be optimised, beta. In this example beta currently = 1 therefore **improvement attempts** = 100 and **improvements** = 10. If the length of the array for beta became overly large the number of **improvement attempts** and **improvements** required to move to the next **step** in the algorithm would soon become overly cumbersome.

When the algorithm has been run the progress is reported in the reporting area as each minor and major step in the algorithm is taken as can be seen below. When the model has been calibrated the outputs are saved and finally the word Done is printed to the screen, it is now safe to close the window.



The outputs are automatically saved in an outputs folder created in the same location as the destination file. The outputs folder will contain:



There are a set of 4 files for the run of the model that had the best fit overall and for the final run of the model. The output files are:

* *Xxxx Flow Matrix.csv*: this is a tabular representation of the outputs with the columns being the destinations (in the same order as the destination file read in) and the rows being the origins (in the same order as the origins file read in).
* *Xxxx Flow Probabilities*.csv: this is a long thin file where each line represents a unique origin-destination pairing. The first column is the origin id the second is the destination id and the third is the probability of a flow occurring between the origin to the destination. All of the probabilities should add to 1 for all destination summed over each origin.
* *Xxxx Matrix*.csv: this file is in the same format as the probabilities file above except that the probability is replaced with the actual flow calculated going from the origin to the destination.
* *Xxxx Stats*.csv: this file contains the distance and fit statistic that were produced for the relevant model run.