**FEAORA Usage Guidelines**

**General Guidelines**

1. **Minimum System Requirements:**

Java Runtime Environment 1.8 +

FICO Xpress Software Suite Version 7.6

MS Excel for viewing excel sheets

MS Internet Explorer in Windows OS for viewing Google Map

Operating systems: Windows 7 (and above) and Mac OS 10.X (and above)

1. **Coding guidelines:**
2. Mosel file should have all headers that needed to be used. For example: Add the **“mmsheet”** or **“mmodbc”** header to work with external excel files from FEAORA tool.
3. Variables should only be declared inside “parameters” and “declaration” blocks, others variables are not shown in the input modeling screen.
4. The variables on which other variables depend should be defined before they are used for assignment of other variables.
5. The variables on which other variables depend should not be taken from external file from GUI as this would result in violation of “mosel” syntax if these are used for size of arrays, list or set.
6. Dependent variables from external file excel range should not depend on a variable that is set from UI as the initialization from excel is done before assignment of normal variables via step, randomized or manual input. Instead define the external excel file related variable directly in “mosel”, you can then redefine the variable in the range field via manual, randomized or step input.
7. All the “mosel” files that you want to import in FEAORA, should be kept inside **“Models”** folder.
8. External data source files like excel should be kept in the **“Models/Data”** folder.
9. Chart Visualizations can only be plotted for singular values and not for dense data types like arrays, sets, lists. So both x and y axis should be singular variables and values from all iterations are taken into consideration. You can apply filter to the values placed left to the chart in the excel file.
10. In map visualization the iteration number starts from 1 and can be found at the corresponding row of “output” tab in excel solution file for that problem.
11. In map visualization different map types have different requirements:
    1. **PointsOnly** map should be used with a single dimensional array as source and a single dimensional array as node condition or show all nodes must be selected.
    2. **PointsWithPath** / **PointsWithDirectedPath** map should be used with two single dimensional array as source and destination and a two dimensional array as edge or path condition between source and destination.

## Example ‘Mosel’ Code

This section provides sample ‘‘Mosel’’ code involving a knapsack problem and some code for demonstrating map visualization.

model Burglar11

uses "mmxprs"

declarations

SIZE = 10

WTMAX = 100 ! Maximum weight allowed

ITEMS = 1..SIZE

VALUE: array(ITEMS) of real ! Value of items

WEIGHT: array(ITEMS) of real ! Weight of items

MaxVal: linctr

Depots = {"Delhi","Mumbai","Munich"}

! decision variable which depots have been selected

NodeCondition: array(Depots) of integer

! decision variable which paths have been selected

EdgeCondition: array(Depots,Depots) of integer

end-declarations

declarations

take: array(ITEMS) of mpvar ! 1 if we take item i; 0 otherwise

end-declarations

! just for displaying map usage

NodeCondition::(["Delhi","Mumbai","Munich"])[0,1,1]

EdgeCondition::[0,1,1,0,0,0,0,1,0]

! Objective: maximize total value

MaxVal:= sum(i in ITEMS) VALUE(i)\*take(i)

! Weight restriction

sum(i in ITEMS) WEIGHT(i)\*take(i) <= WTMAX

! All variables are 0/1

forall(i in ITEMS) take(i) is\_binary

maximize(MaxVal) ! Solve the MIP-problem

! Print out the solution

writeln("Solution:")

forall(i in ITEMS) writeln(" take(", i, "): ", getsol(take(i)))

! Output solution to calling application

!initializations to IODRV

! evaluation of getobjval as SOLVAL

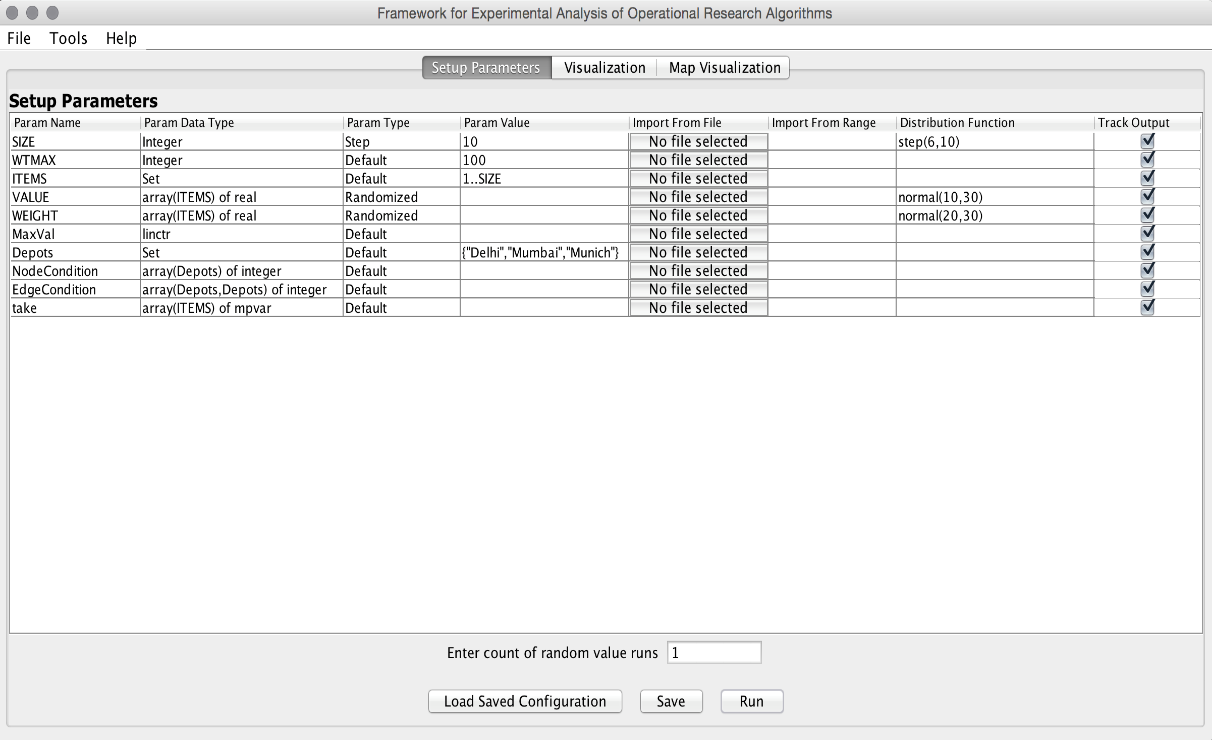
! evaluation of round(sum(i in ITEMS) getsol(take(i))) as NUM

!end-initializations

end-model

## Sample Input Configuration

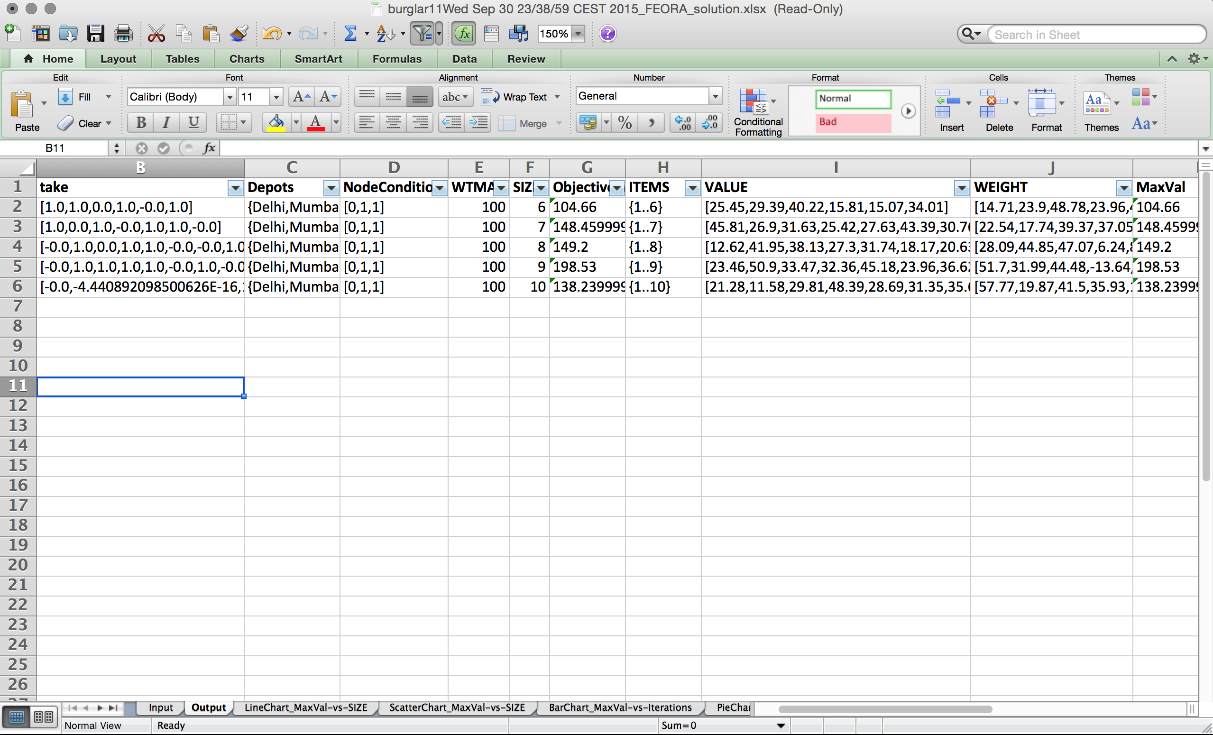
The sample input configuration for the above problem is shown below. All the variables are marked for output tracking.



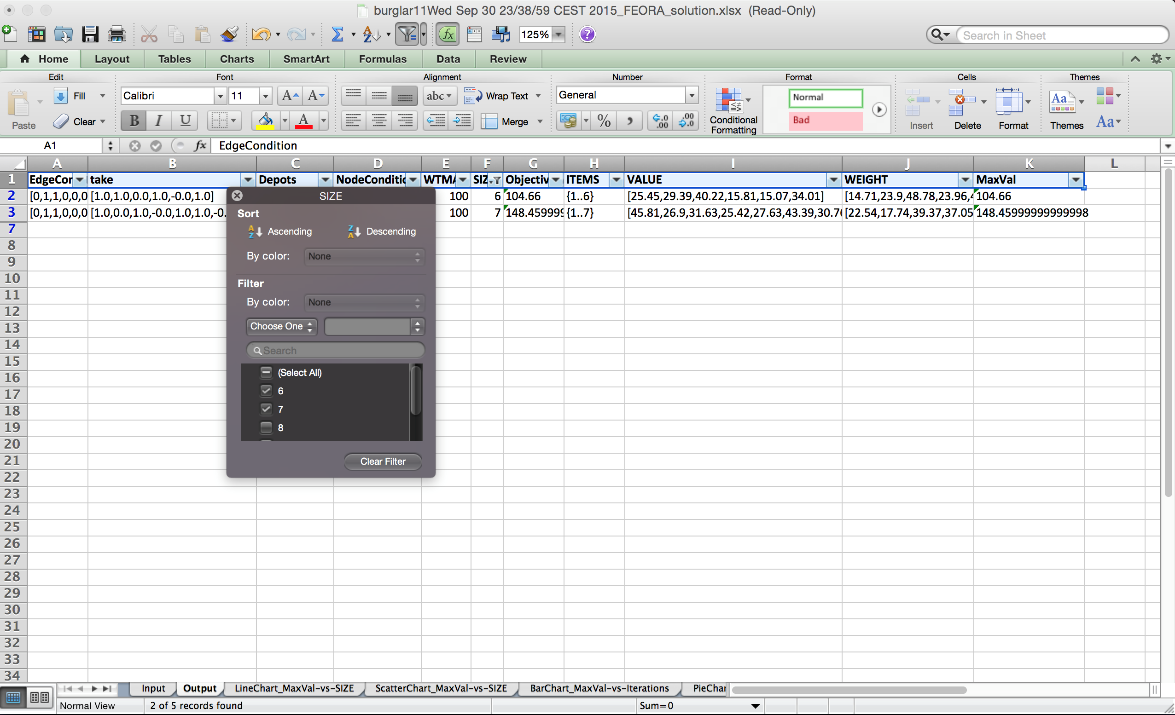
**Sample input configuration screen**

## Output analysis

The tracked variables of a model execution gets saved in an excel file in the Models folder with a timestamp in file name. User can easily filter or sort the results using excel filtering and sorting functionality. This does not require any additional programming skill from the user. Figure below shows the output for the executions.



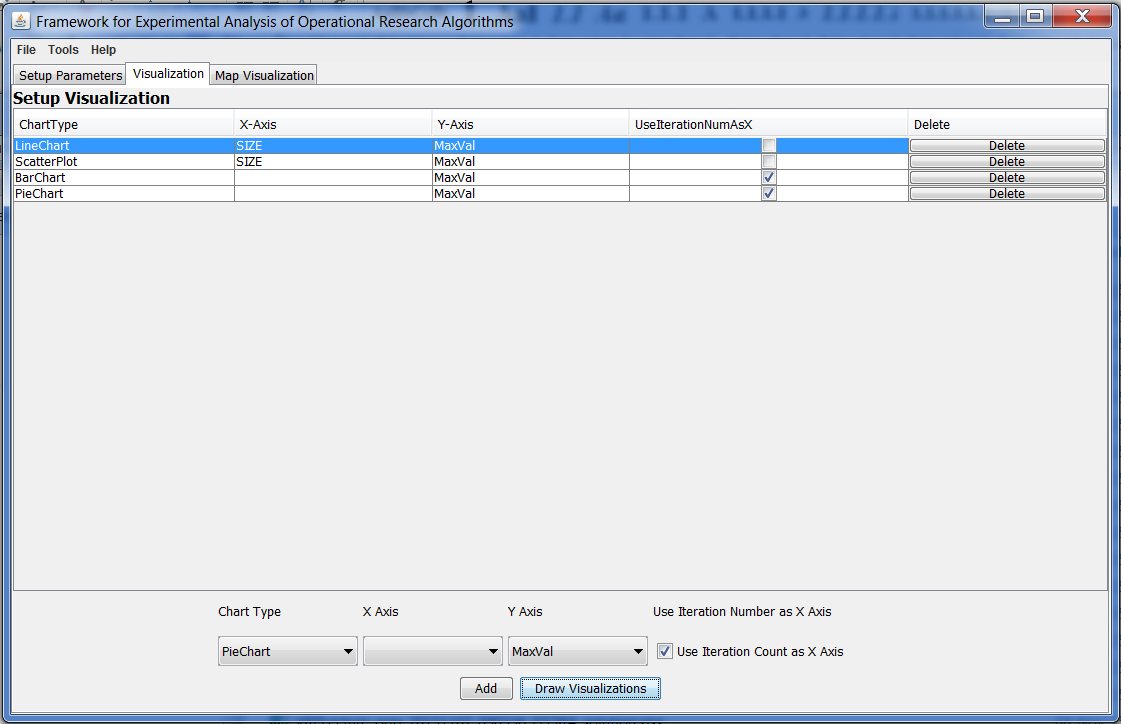
**Output saved in excel format**



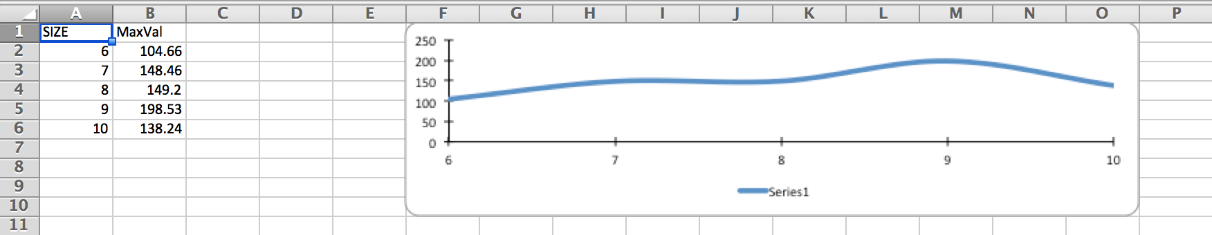
**Filtering of the result rows using excel filter option**

## Sample Chart Visualization Configuration

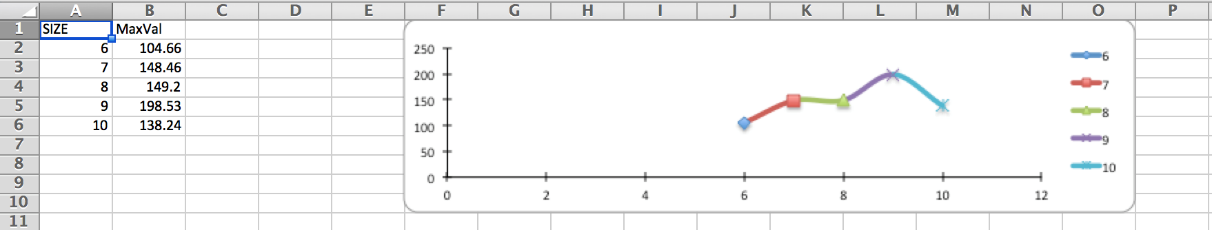
The following figure shows the sample chart configuration for this problem.



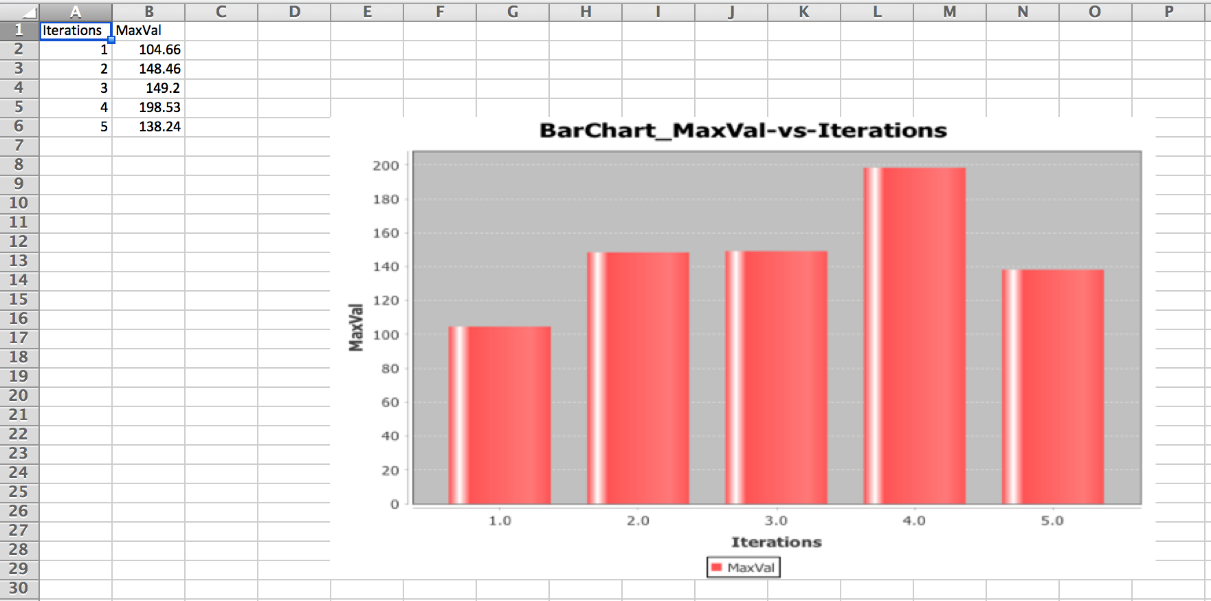
**Sample chart visualization configuration**



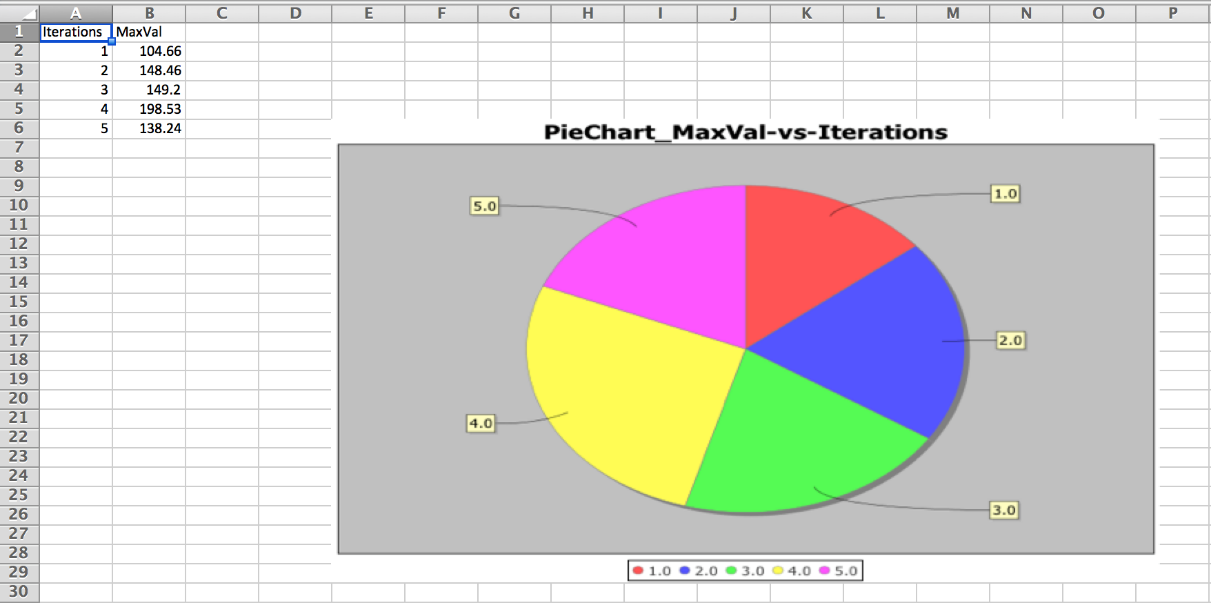
**Sample line chart visualization**



**Scatter chart visualization**

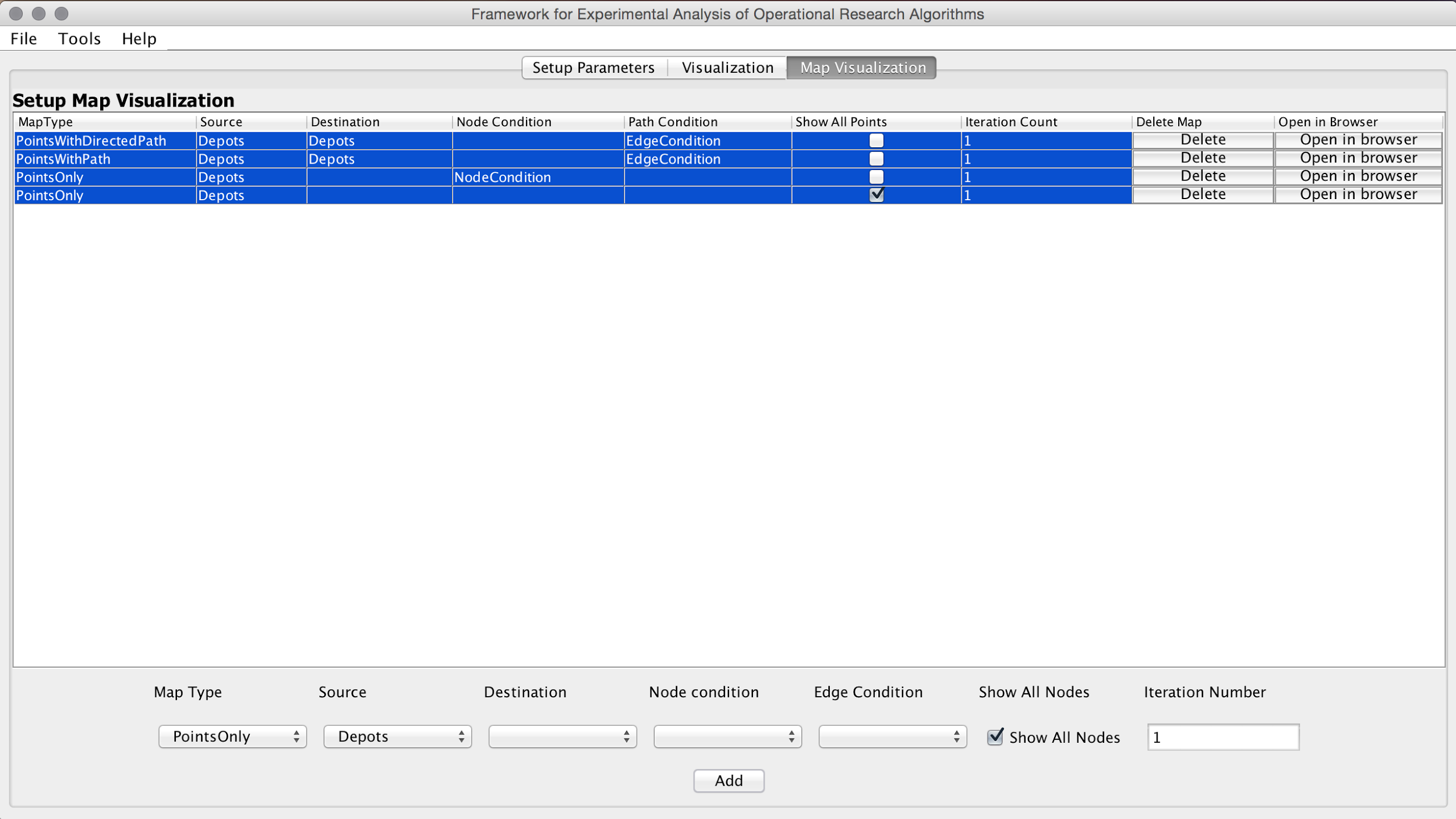


**Bar chart visualization**



**Pie chart visualization**

## Sample Map Visualization Configuration



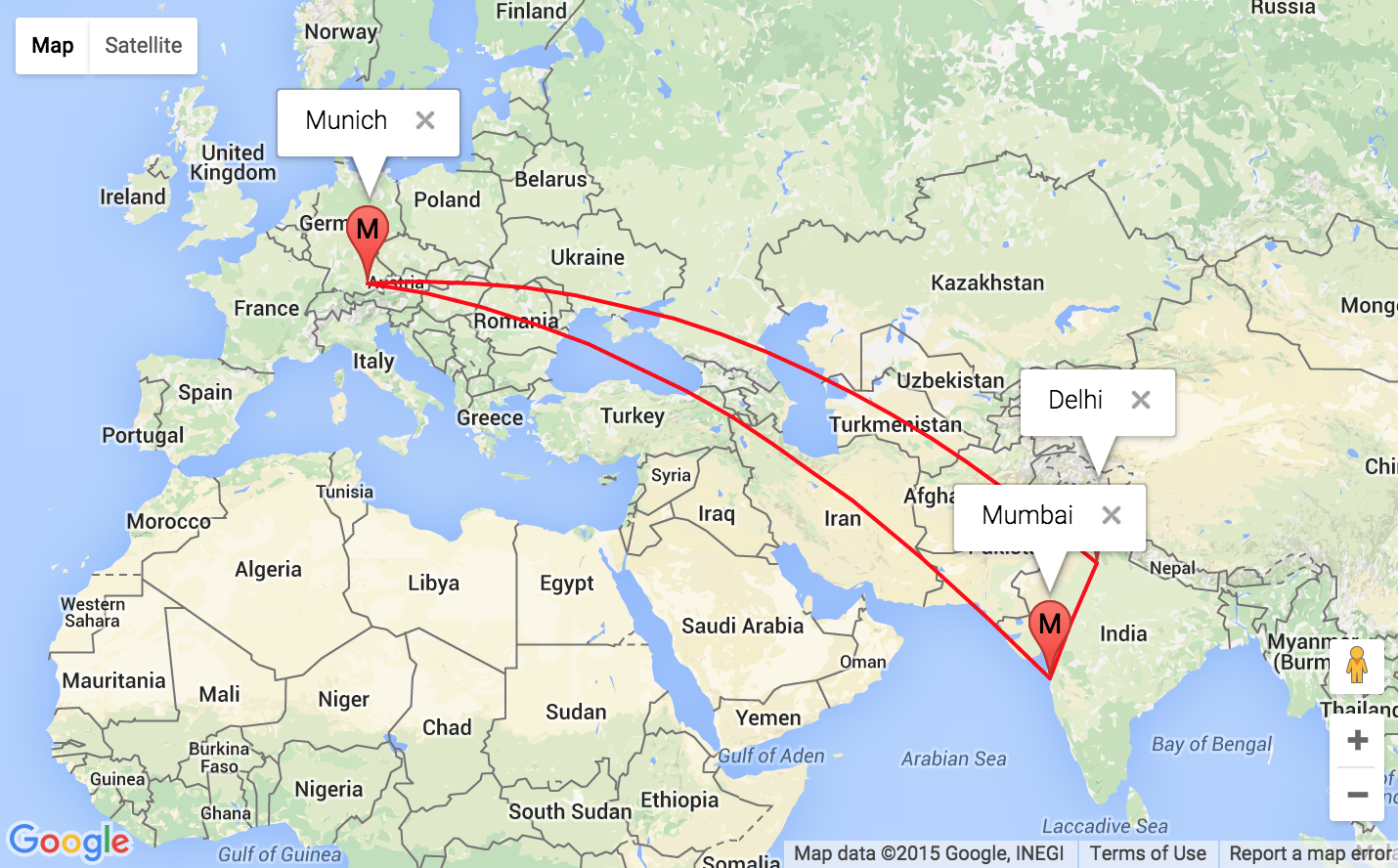
**Sample map visualization configuration**



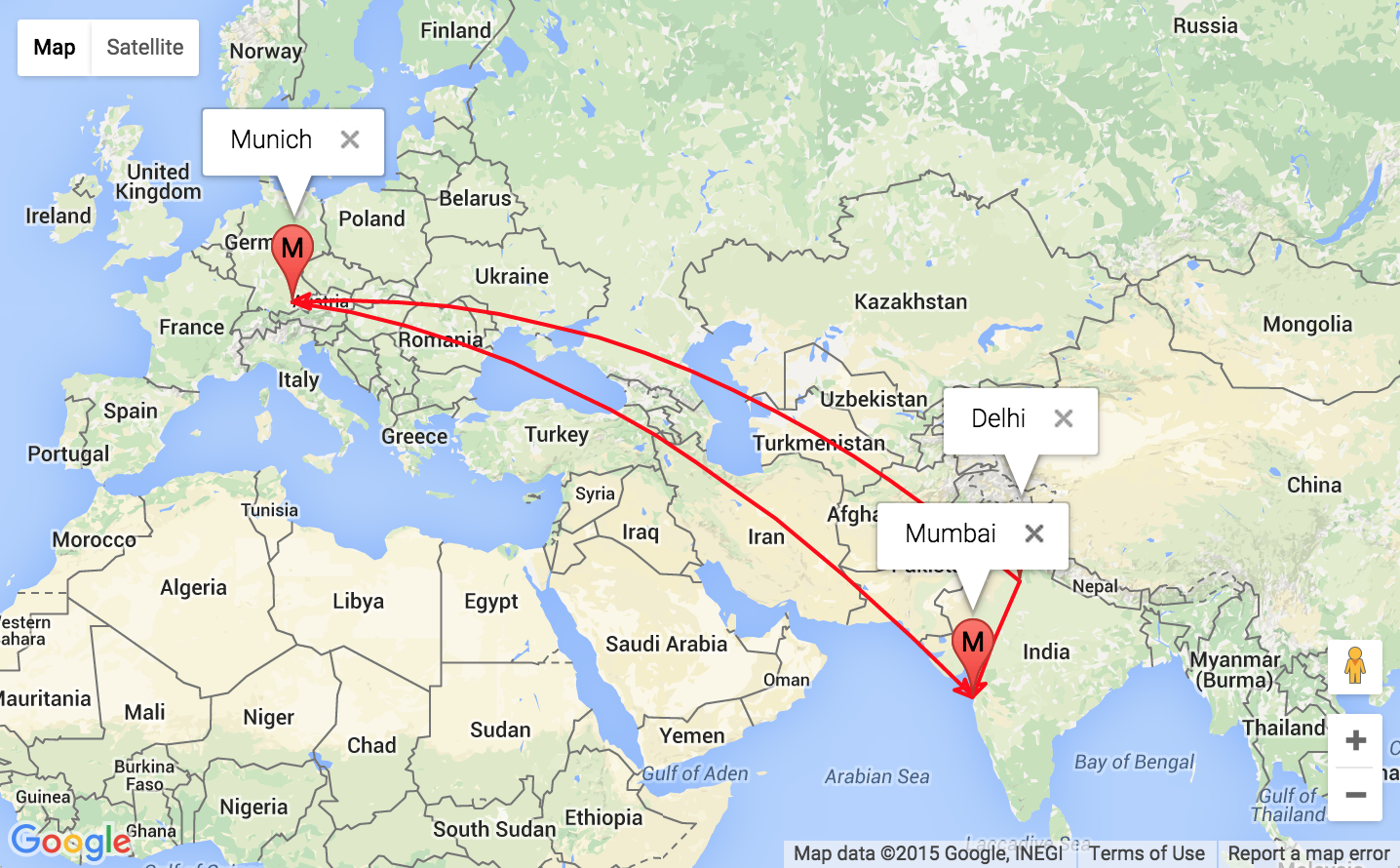
**Points only map output with show all nodes option checked**



**Points only map with filtered points using a 1-D node condition array**



**Points with path map with filtered edge using 2-D edge condition array**



**Points with directed paths with filtered edge using 2-D edge condition array**

## Example VRP (Vehicle Routing Problem) ‘Mosel’ Code

This section provides sample ‘‘Mosel’’ code involving a VRP problem and some code for demonstrating map visualization and excel input

## Sample Input Configuration

The sample input configuration for the above problem is shown below. All the variables are marked for output tracking. (Models\VRPLectureModified.mos)

model VRP

uses "mmxprs", "mmsheet"; !gain access to the Xpress-Optimizer solver

declarations

!Sets

n: integer !number of customers

N: set of integer !set of nodes

C: set of integer !set of customer

!Parameters

X: array(N) of integer !x-position of nodes

Y: array(N) of integer !Y-position of nodes

d: array(N, N) of real ! distance matrix

Nodes: array(N) of string !Location name of nodes

masterDist: array(0..10, 0..10) of real !master distance matrix

q: array(C) of integer ! demand

cap: integer ! capacity of a vehicle

M: integer ! big M

!dv's

x: array(N, N) of mpvar ! driving from i to j

y: array(N) of mpvar ! vehicle capacity at j

!labels

dist: linctr !objective function

ctrLeave: array(C) of linctr

ctrFlow: array(N) of linctr

ctrCapA: array(C) of linctr

ctrCapB: array(C) of linctr

ctrLoad: array(C,C) of linctr

end-declarations

N := 0..n

C := 1..n

initializations from "mmsheet.excel:Models//Data//VRP-Data.xlsx"

X as 'noindex;[Daten\_10$B5:B'+(5+n)+']'

Y as 'noindex;[Daten\_10$C5:C'+(5+n)+']'

q as 'noindex;[Daten\_10$D6:D'+(5+n)+']'

cap as 'noindex;[Daten\_10$H2]'

M as 'noindex;[Daten\_10$H1]'

Nodes as 'noindex;[Daten\_10$E5:E'+(5+n)+']' ! use of dynamic variable

masterDist as 'noindex;[Daten\_10$G5:Q15]'

end-initializations

!pre-processing

! use master variable as column name require alphabet which cannot be dynamic

forall(i, j in N)

d(i,j) := masterDist(i,j)

forall(i in C)

ctrLeave(i) := sum(j in N) x(i,j) = 1

forall(i in N)

ctrFlow(i) := sum(j in N) x(i,j) = sum(j in N) x(j,i)

forall(i,j in C) ctrLoad(i,j) := y(j)- y(i) >= q(j) - M\*(1-x(i,j))

forall(i in C) ctrCapA(i) := q(i) <= y(i)

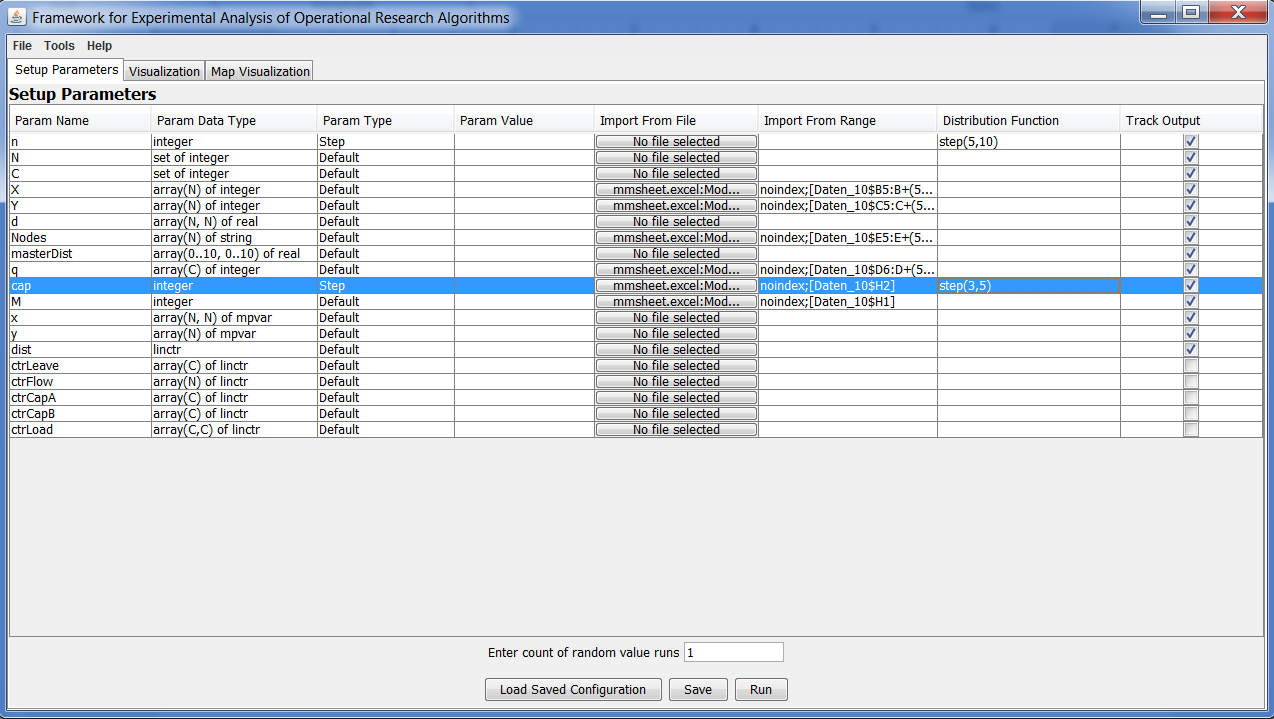
forall(i in C) ctrCapB(i) := y(i) <= cap

forall(i,j in N) x(i,j) is\_binary

dist := sum(i,j in N) d(i,j)\*x(i,j)

minimize(dist)

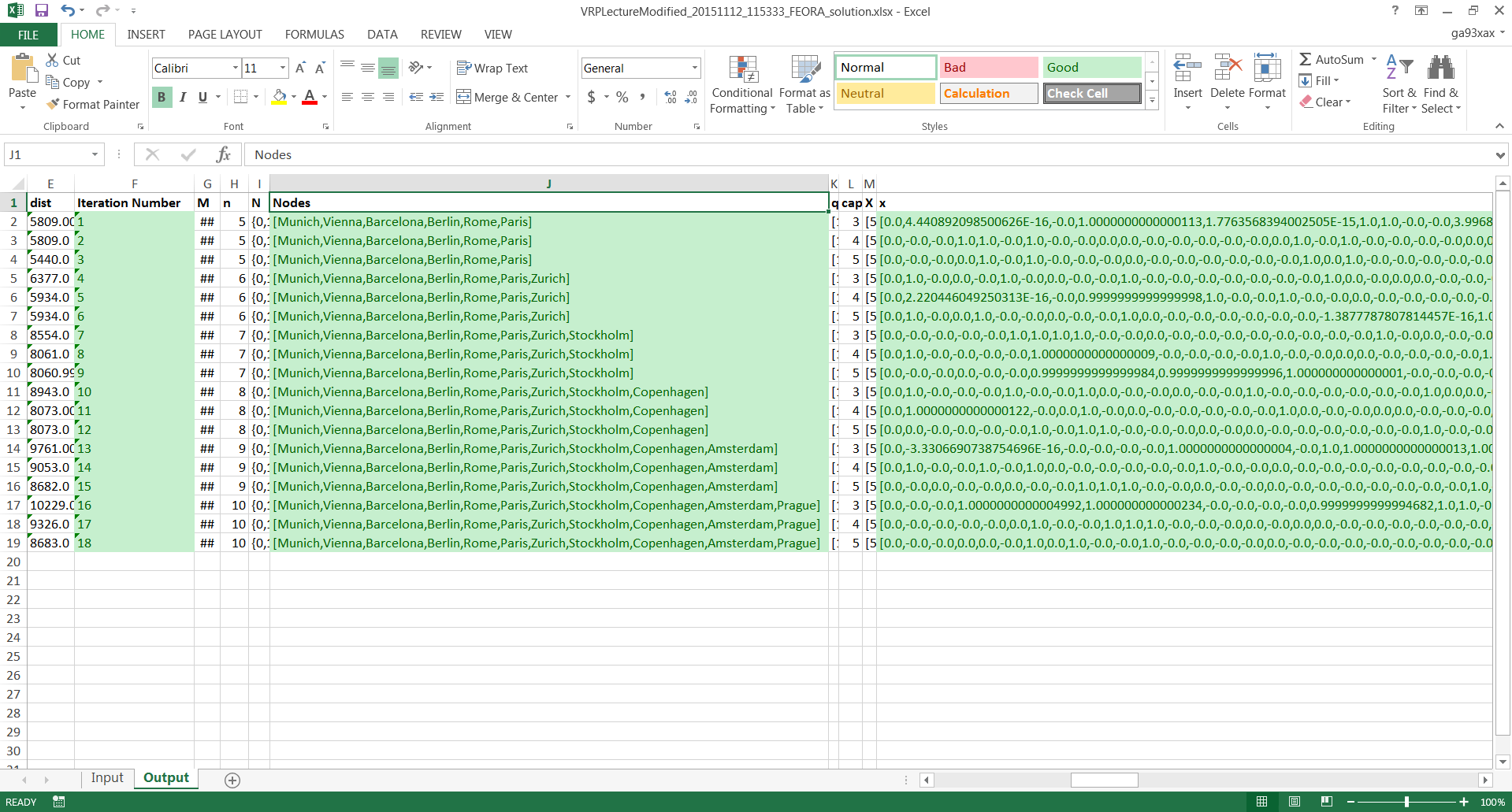
end-model



**Sample input configuration screen**

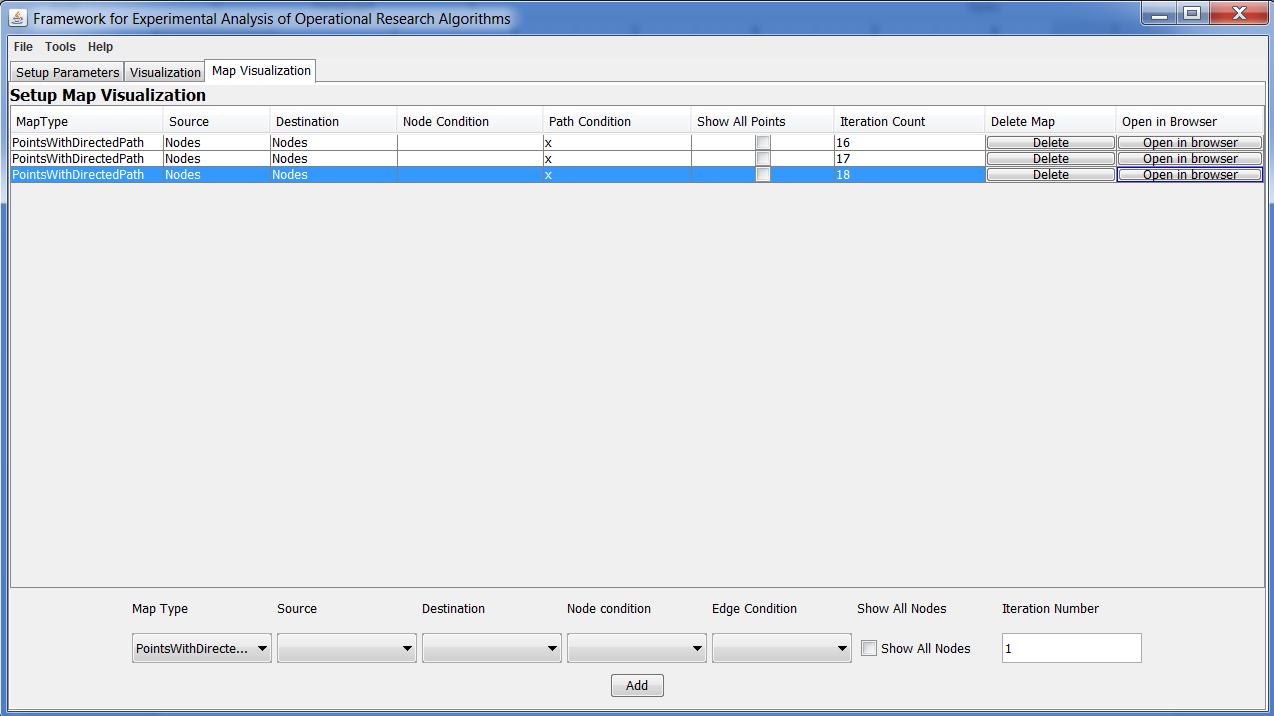
## Output analysis

The tracked variables of a model execution gets saved in an excel file in the Models folder with a timestamp in file name. User can easily filter or sort the results using excel filtering and sorting functionality. This does not require any additional programming skill from the user. Figure below shows the output for the executions. ***Note that the Iteration number for the row is to be used in map visualization panel to load resultant Edge-Condition matrix.***

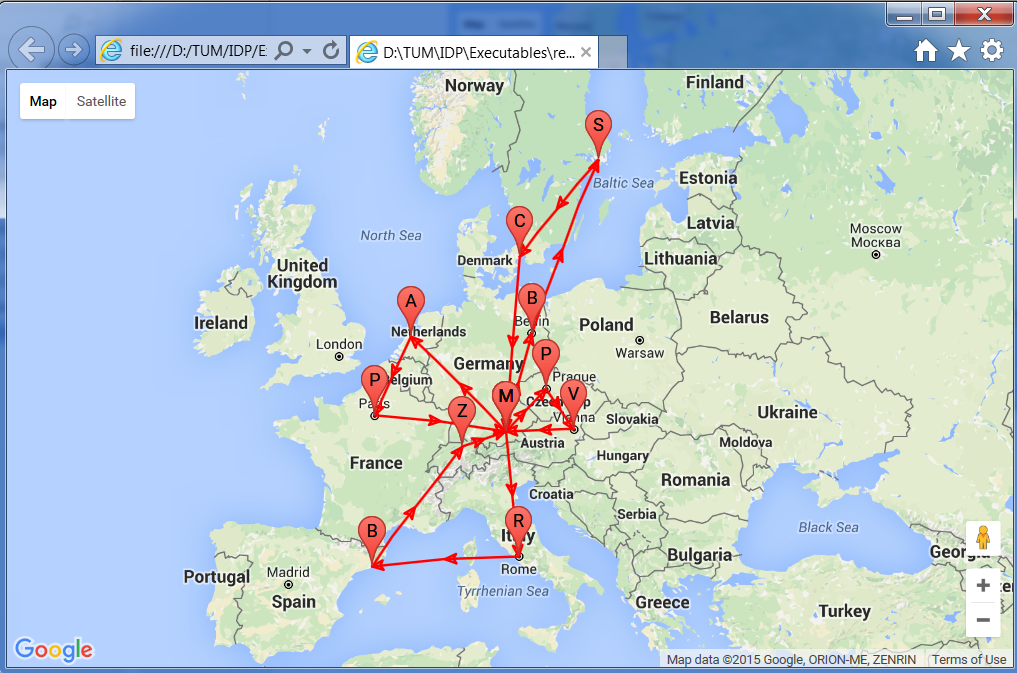


**Output saved in excel format**

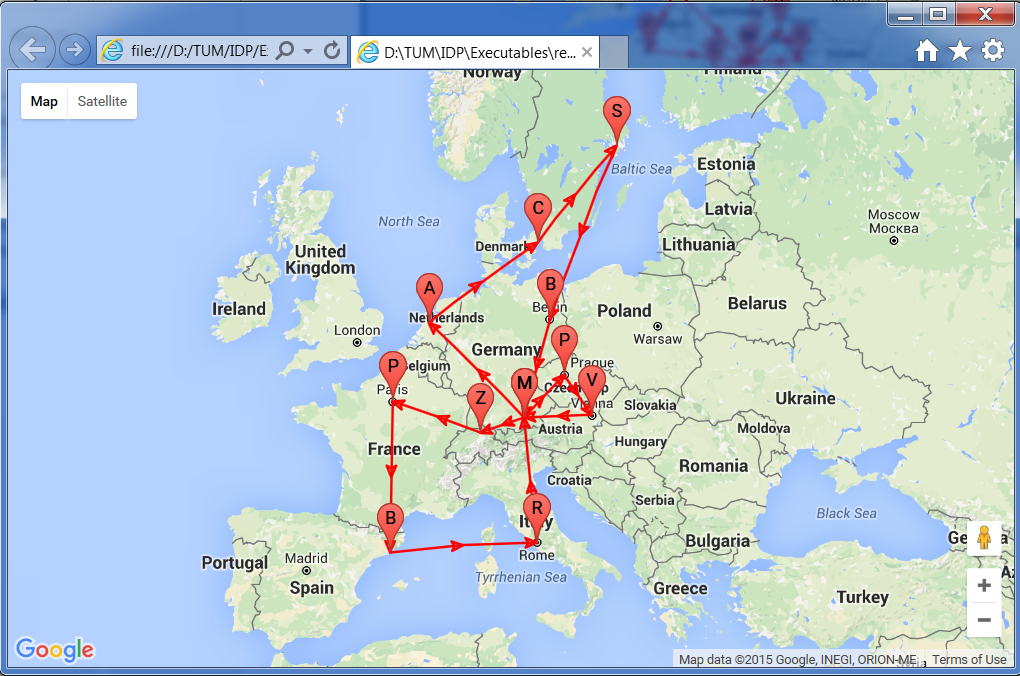
## Sample Map Visualization Configuration



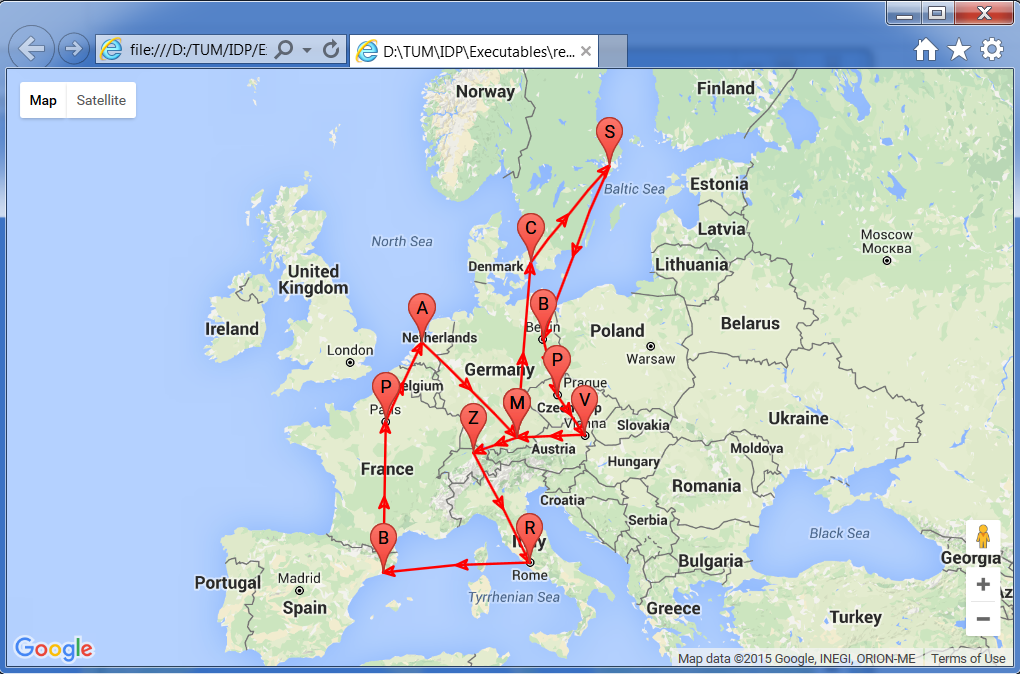
**Sample map visualization configuration**



**Result of iteration number 16 with vehicle capacity of 3**



**Result of iteration number 17 with vehicle capacity of 4**



**Result of iteration number 18 with vehicle capacity of 5**

### Probability Distributions Functions

Following probability distributions are supported by FEAORA framework. These are to be used with randomized option for ‘Param Type’ and will generate array or value depending on what kind of parameter it is.

#### http://www.anylogic.com/anylogic/help/topic/com.xj.anylogic.help/html/functions/images/exponential.pngExponential

The Exponential distribution is a continuous distribution bounded on the lower side. Its shape is always the same, starting at a finite value at the minimum and continuously decreasing at larger x. The Exponential distribution decreases rapidly for increasing x.

The Exponential distribution is frequently used to represent the time between random occurrences, such as the time between arrivals at a specific location in a queuing model or the time between failures in reliability models. It has also been used to represent the services times of a specific operation. Further, it serves as an explicit manner in which the time dependence on noise may be treated. As such, these models are making explicit use of the lack of history dependence of the exponential distribution; it has the same set of probabilities when shifted in time. Even when Exponential models are known to be inadequate to describe the situation, their mathematical tractability provides a good starting point.

**Sample:** lambda=1;



Exponential distribution sample curve

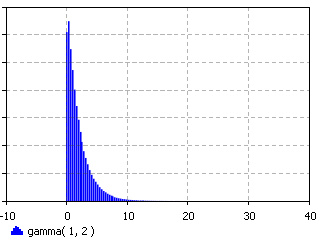
**Usage:**

* exponential(lambda)
* exponential(): Assumes lambda = 1

#### http://www.anylogic.com/anylogic/help/topic/com.xj.anylogic.help/html/functions/images/gamma.pngGamma

The Gamma distribution is a continuous distribution bounded at the lower side. It has three distinct regions. For alpha=1, beta=1/lambda the Gamma distribution reduces to the exponential (lambda) distribution, starting at a finite value at minimum x and decreasing monotonically thereafter. For alpha<1, the Gamma distribution tends to infinity at minimum x and decreases monotonically for increasing x. For alpha>1, the Gamma distribution is 0 at minimum x, peaks at a value that depends on both alpha and beta, decreasing monotonically thereafter. If alpha is restricted to positive integers, the Gamma distribution is reduced to the Erlang distribution. The Gamma distribution has been used to represent lifetimes, lead times, personal income data, a population about a stable equilibrium, inter-arrival times, and service times. In particular, it can represent lifetime with redundancy.

Note that in FEAORA implementation min is always 0.  
**Sample:** gamma(1,2)



Gamma distribution sample curve

**Usage:**

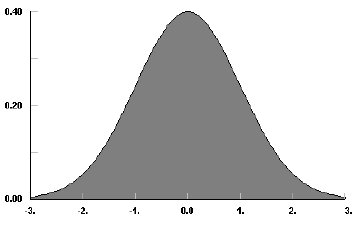
* gamma(alpha, beta)

#### Normal

|  |  |
| --- | --- |
| *Density* | http://www.anylogic.com/anylogic/help/topic/com.xj.anylogic.help/html/functions/images/normal_density.png |
| *Mean* | http://www.anylogic.com/anylogic/help/topic/com.xj.anylogic.help/html/functions/images/normal_mean.png |
| *Variance* | http://www.anylogic.com/anylogic/help/topic/com.xj.anylogic.help/html/functions/images/normal_variance.png |
| *Mode* | http://www.anylogic.com/anylogic/help/topic/com.xj.anylogic.help/html/functions/images/normal_mean.png |

The Normal distribution is an unbounded continuous distribution. It is sometimes called a Gaussian distribution or the bell curve. Because of its property of representing an increasing sum of small, independent errors, the Normal distribution finds many, many uses in statistics. It is wrongly used in many situations. Possibly, the most important test in the fitting of analytical distributions is the elimination of the Normal distribution as a possible candidate.   
The Normal distribution is used as an approximation for the Binomial distribution when the values of n, p are in the appropriate range. The Normal distribution is frequently used to represent symmetrical data, but suffers from being unbounded in both directions. If the data is known to have a lower bound, it may be better represented by suitable parameterization of the Gamma distributions. If the data is known to have both upper and lower bounds, the Beta distribution can be used, although much work has been done on truncated Normal distributions.

**Sample:** sigma =1; mean = 0



Normal distribution sample curve

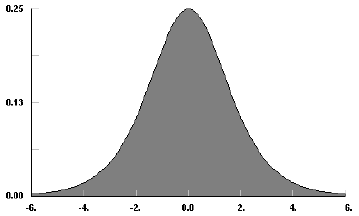
**Usage:**

* normal(sigma, mean)
* normal(sigma): mean is assumed to be 0
* normal(): sigma is 1, mean is 0

#### http://www.anylogic.com/anylogic/help/topic/com.xj.anylogic.help/html/functions/images/logistic.pngLogistic

The Logistic distribution is an unbounded continuous distribution which is symmetrical about its mean [and shift parameter], alpha. The shape of the Logistic distribution is very much like the Normal distribution, except that the Logistic distribution has broader tails.   
  
The Logistic function is most often used a growth model: for populations, for weight gain, for business failure, etc. The Logistic distribution can be can be used to test for the suitability of such a model, with transformation to get back to the minimum and maximum values for the Logistic function. Occasionally, the Logistic function is used in place of the Normal function where exceptional cases play a larger role.

**Sample:** beta = 1; alpha =0

**

Logistic distribution sample curve

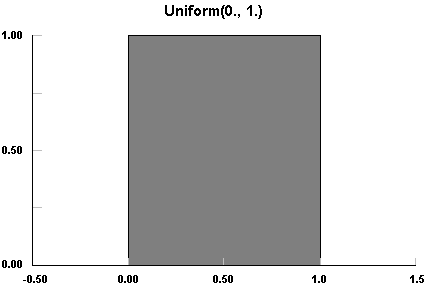
**Usage:**

* logistic(beta, alpha)

#### http://www.anylogic.com/anylogic/help/topic/com.xj.anylogic.help/html/functions/images/uniform.pngUniform

The Uniform distribution is a continuous distribution bounded on both sides, i.e. the sample lays in the interval [min,max). The probability density does not depend on the value of x. It is a special case of the [Beta](http://www.anylogic.com/anylogic/help/topic/com.xj.anylogic.help/html/functions/beta.html) distribution. It is frequently called rectangular distribution.   
The Uniform distribution is used to represent a random variable with constant likelihood of being in any small interval between min and max. Note that the probability of max value is 0; the max point never occurs.

**Sample:** uniform(0,1)



Uniform distribution sample curve

**Usage:**

* uniform(): same as uniform(0,1)
* uniform(max): same as uniform(0,max)
* uniform(min, max)

### Step Distribution

Unlike random distribution the step distribution is not at all random but allocates samples sequentially between the given range of integer value. A step variable can be given to jump particular number of integers. After sample count is overflowed the sample value is again set to min.

**Sample:** step(1,7,2)

Normal distribution sample curve: The chart shows the sampled value after each sample count on x-axis

**Usage:**

* step(min, max, step): Allocates samples sequentially between and inclusive of min and max values and next sample is always current sample + the step value. Therefore it can result in exclusion of max value if the addition of step value to the penultimate sample results in overflow.
* step(min, max):Uses default step as 1.
* step(max):Uses default min as 1 and default step as 1.