

AGENT SYSTEMS – PROJECT

USING AGENT SYSTEMS TO ESTIMATE THE RELIABILITY OF TIME -
CONSTRAINED MULTI-STATE FLOW NETWORKS WITH CORRELATED
FAULTS AND NON-INTEGER CAPACITIES

Paweł Kozyra
GUT – FETI

Introduction

The goal of the project is to implement and test a multi-agent system to estimate the reliability of a time-constrained multi-state flow network with correlated faults and non-integer capacities. For the entire project, you can obtain 50 points. The point distribution is indicated in parentheses using blue font. The project should consist of 4 classes:

1. MFN
2. SSVGenerator extends Agent
3. SSVGeneratorGui extends JFrame
4. TT extends Agent

Description of the classes

MFN (25)

IMPORTANT! This course focuses on agent systems. Therefore, your implementation of methods in this class will be graded if and only if they are used by other methods in the MFN class or by at least one of the agents: the SSVGenerator agent or the TT agent. In other words, an unused method will receive a score of 0 points.

The description of this class can be found in [1] P. M. Kozyra, "A dictionary algorithm for the solution to the generalized quickest path reliability problem", Reliability Engineering and System Safety, pp. 8–11. The implemented model is simplified and does not require modeling the network topology, since the lists of all minimal paths, MP_{s0} and MP_{s1}, for given multistate flow networks are provided. In particular, the MFN class should contain the fields necessary to implement the task: (1)

- the number of links - int m
- the component number vector -int[] W,
- the component capacity vector - double[] C,
- the lead time vector - int[] L,
- the component reliability vector - double[] R,
- the vector of the correlation between the faults of the components - double[] rho
- the beta vector– double[] beta – described by formula (2) from [1];
- the list of minimal paths - ArrayList<int[]> MPs

The MFN class should contain the inner class Combinatorial implementing methods needed to compute factorial $n!$ and binomial coefficient $\binom{n}{k}$. (2)

The constructor of the MFN class should do the following: (2)

- check whether the length of vectors W, C, L, R, and rho is equal to m;
- check whether all values of R and rho are between 0 and 1;

- create the beta vector if the above-mentioned conditions are satisfied.

The MFN class should implement methods defined by formulae (1), (3) - (5), and (8) from [1]. (5)

Apart from this, the MFN class should also implement additional methods:

- void getMPs(String fileName) that reads the file with the file name = filename and creates ArrayList<int[]> MPs; (3)
- double[][] CDF(double[][] arPMF) that creates an array of values of the cumulative distribution function based on an array arPMF created by formula (1); (1)
- static double normalCDF(double z) that computes an approximated value of the cumulative distribution function of the standard normal distribution for n=100, based on the formula (https://en.wikipedia.org/wiki/Normal_distribution)

$$\Phi(x) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \cdot e^{-x^2/2} \left[x + \frac{x^3}{3} + \frac{x^5}{3 \cdot 5} + \dots + \frac{x^{2n+1}}{(2n+1)!!} + \dots \right]$$

where !! denotes the double factorial and should also be implemented. (2)

- static double normalICDF(double u) that computes an approximated value of the quantile function (the inverse of the cumulative distribution function) of the standard normal distribution

IMPORTANT! In order to implement normalICDF, invent your own algorithm that for given value u, it determines a real number x such that

$$|normalCDF(x) - u| \leq 10^{-1} \quad (5)$$

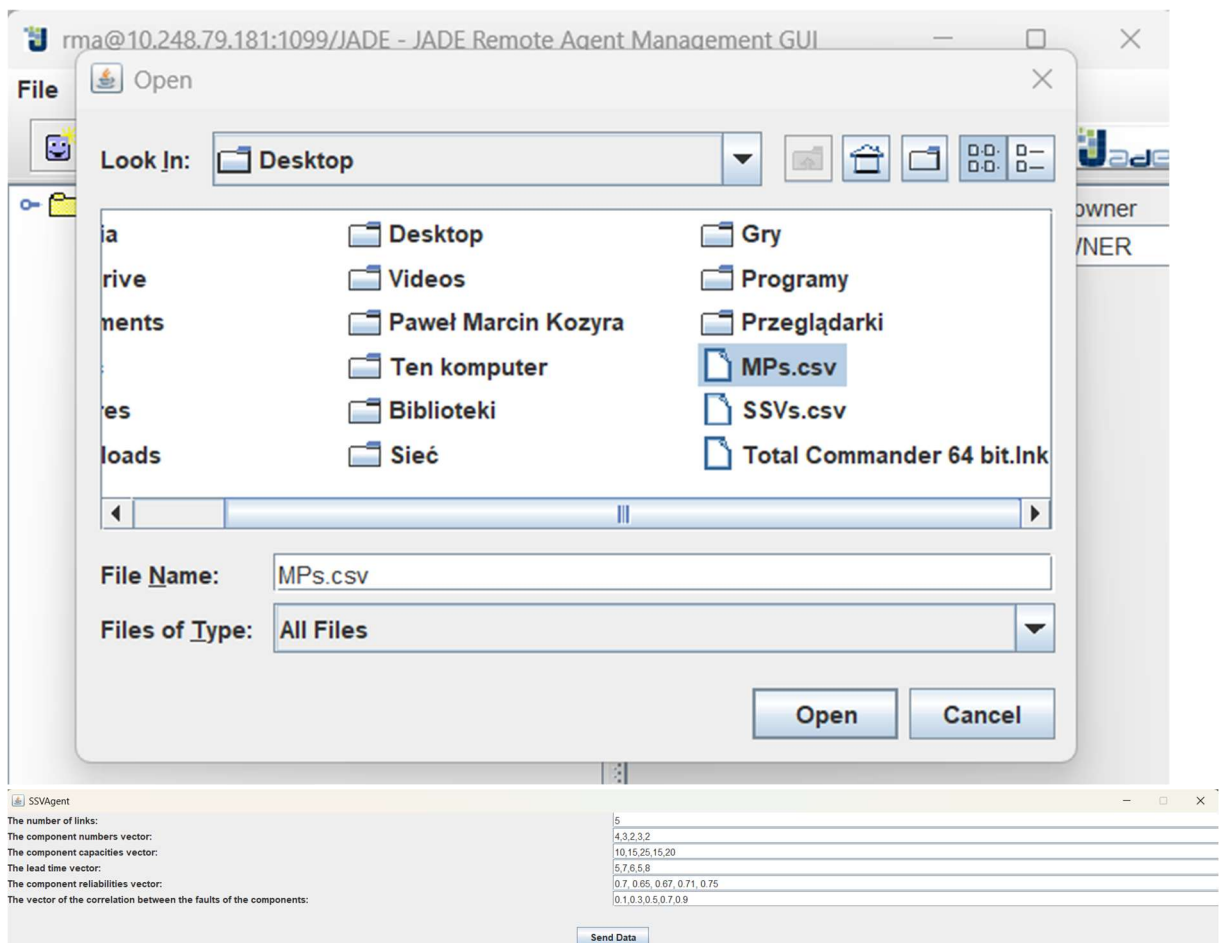
- Based on formula (12b) (from [2] G.S. Fishman – “Monte Carlo, Concepts, Algorithms, and Applications” – Springer), implement a function finding the worst-case normal sample size; (1)
- Based on the inverse CDF method applied to discrete distribution or the Chen and Asau Guide Table Method coming from [3] J. E. Gentle – “Random Number Generation and Monte Carlo Methods” – Springer (2005), implement method double[][] randomSSV(int N, double[][]arCDF), that, for a given integer N and an array arCDF of values of the cumulative distribution function, generates N random system state vectors (SSVs). (3)

SSVGenerator and SSVGeneratorGui (17)

The SSVGenerator class should implement an agent performing the following actions:

- During activation of the agent in cmd you should pass as its arguments values ϵ and δ in order to determine the number N of SSVs based on formula (12b) from [2]. The agent should check whether these values are between 0 and 1 (if not, the agent should terminate). (2)

- After activation, the agent should show the GUI of SSVGeneratorGui in which you should be able to choose a .csv file with MPs and specify the parameters of an MFN (5):



- After clicking the “Send Data” button, the agent should create an object mfn of the MFN class and display the parameters of this MFN, such as W, C, L, R, and rho. (1)
- Next, based on the randomSSV method, the agent should generate N random SSVs. (0.5)
- Next, the agent should look for the TT agent responsible for the computation of transmission times and network reliability estimation. (0.5)
- Next, the agent should send to the TT agent a message containing the MFN parameters, the path to the .csv file with MPS, and the generated random SSVs. (6)
- Finally, the agent should receive a message from the TT agent containing the network reliability estimated by the TT agent, display this message, and terminate. (2)

TT (8)

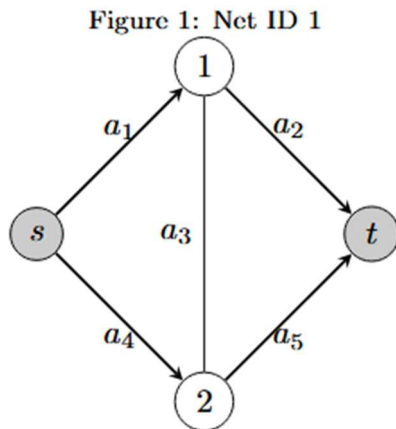
The TT class implements an agent which will be responsible for the transmission times computation and the network reliability estimation based on the information obtained from the SSVGenerator agent. The TT agent should behave as follows:

- During activation, the TT agent should have two arguments (both of double type): the number of units of flow – d , and the maximum transmission time – T . (1)
- The ultimate aim of the TT agent is to estimate the network reliability at level (d,T) , i. e., the probability of sending d units of flow within time T and send this value to the SSVGenerator agent.
- However, before performing the abovementioned action, the TT agent should be able to receive the message from the SSVGenerator agent. Based on this information, the TT agent should write the random system state vectors generated and sent by the SSVGenerator agent to the SSV.csv file. (6)
- Next, based on the MFN method, implementing formula (8) from [1], the TT agent, for each system state vector X , should compare the value of $T(d, X)$ with a given maximum transmission time T . (0.5)
- Based on the aforementioned comparisons, the TT agent should approximate the network reliability at level (d,T) . (0.5)

Examples

Network 1

Let us consider the following network from Fig. 1.



Its parameters are as follows:

$$W = [4, 3, 2, 3, 2]$$

$$C = [10, 15, 25, 15, 20]$$

$$L = [5, 7, 6, 5, 8]$$

$$R = [0.7, 0.65, 0.67, 0.71, 0.75]$$

$$\rho = [0.1, 0.3, 0.5, 0.7, 0.9]$$

The network has 4 minimal paths from s to t :

$P1 = [a1, a2]$, $P2 = [a1, a3, a5]$, $P3 = [a4, a3, a2]$, $P4 = [a4, a5]$.

They are written down in the MPs0.csv file.

We are interested in the estimation of the network reliability at level (42,15.5), i.e., we want to estimate the probability of sending 42 units of flow from the source s to the sink t within time 15.5 through one of 4 minimal paths. The exact network reliability 0.8945941587830153 has been determined using Kozyra's dictionary algorithm [1] and the serial bounding algorithm described in the article DOI: 10.1016/j.res.2021.107500. Based on the SSVs generated by the SSVGenerator agent and written by TT agent down to the SSVs0.csv file, the network reliability estimated by the TT agent is equal to **0.8953460332770677**. So, it differs from the exact reliability value by less than 0.01. If we repeated this process N times (for N tending to infinity), the number of cases where the approximated reliability differs from the exact one more than 0.01 would be approximately 0.01, since we have passed parameters $\epsilon = 0.01, \delta = 0.01$ to the SSVGenerator agent.

After activation agents in cmd you should obtain similar results and SSVs0.csv file:

```
java -cp JADE-all-4.6.0\JADE-bin-4.6.0\jade\lib\jade.jar;classes
```

```
jade.Boot -gui -agents
```

```
SSVAgent:SSVGenerator(0.01,0.01);TTAgent:TT(42,15.5)
```

```
lis 15, 2025 4:53:11 PM jade.core.Runtime beginContainer
```

```
INFO: -----
```

```
    This is JADE 4.6.0 - revision 6869 of 30-11-2022 14:47:03
```

```
    downloaded in Open Source, under LGPL restrictions,
```

```
    at http://jade.tilab.com/
```

```
-----
```

```
lis 15, 2025 4:53:11 PM jade.imtp.leap.LEAPIMTPManager initialize
```

```
INFO: Listening for intra-platform commands on address:
```

```
- jicp://10.248.79.181:1099
```

```
lis 15, 2025 4:53:11 PM jade.core.BaseService init
```

```
INFO: Service jade.core.management.AgentManagement initialized
```

```
lis 15, 2025 4:53:11 PM jade.core.BaseService init
```

```
INFO: Service jade.core.messaging.Messaging initialized
```

```
lis 15, 2025 4:53:11 PM jade.core.BaseService init
```

INFO: Service jade.core.resource.ResourceManagement initialized

lis 15, 2025 4:53:11 PM jade.core.BaseService init

INFO: Service jade.core.mobility.AgentMobility initialized

lis 15, 2025 4:53:11 PM jade.core.BaseService init

INFO: Service jade.core.event.Notification initialized

lis 15, 2025 4:53:11 PM jade.mtp.http.HTTPServer <init>

INFO: HTTP-MTP Using XML parser

com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl\$JAXPSAXParser

lis 15, 2025 4:53:11 PM jade.core.messaging.MessagingService boot

INFO: MTP addresses:

http://LAPTOP-T02T5KC7:7778/acc

lis 15, 2025 4:53:11 PM jade.core.AgentContainerImpl joinPlatform

INFO: -----

Agent container Main-Container@10.248.79.181 is ready.

Hallo! Transmission times computing-agent

TTAgent@10.248.79.181:1099/JADE is ready.

The aim is to estimate the probability of sending 42.0 units of flow within time 15.5

Hallo! SSVGenerator-agent SSVAgent@10.248.79.181:1099/JADE is ready.

The minimum number of iterations is equal to 16588

It has been created the MFN with the following parameters:

W=[4,3,2,3,2]

C=[10.0,15.0,25.0,15.0,20.0]

L=[5,7,6,5,8]

R=[0.7,0.65,0.67,0.71,0.75]

rho=[0.1,0.3,0.5,0.7,0.9]

16588 random SSVs have been generated!

Found the following transmission times computing agent:

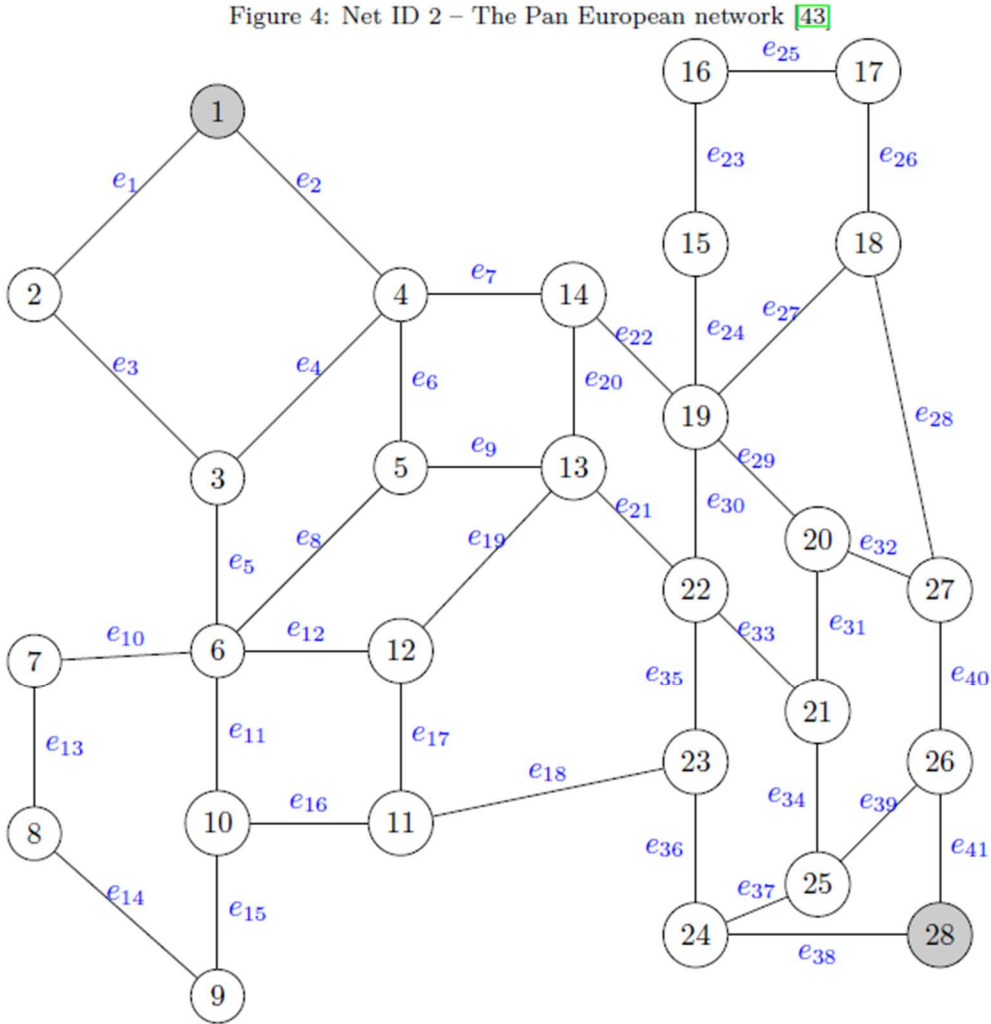
TTAgent@10.248.79.181:1099/JADE

Estimated network reliability is equal to 0.8953460332770677

SSVGenerator-agent SSVAgent@10.248.79.181:1099/JADE terminating.

Network 2

Let us consider the following network.



Its parameters are as follows:

$m=41$

$W = [43, 32, 41, 19, 15, 12, 29, 27, 20, 11, 45, 25, 25, 37, 46, 17, 26,$
 $16, 28, 27, 33, 30, 47, 23, 31, 10, 20, 28, 31, 36, 12, 44, 26, 41,$
 $38, 16, 21, 28, 35, 37, 42]$

$C = [6.25, 14.65, 9.55, 13.95, 12. , 6.5 , 6.89, 9.36, 13.72,$
 $7.9 , 6.66, 8.14, 13.84, 10.53, 7.28, 8.53, 8.92, 10.22,$
 $5.39, 7.09, 12.2 , 11.32, 10.15, 7.7 , 11.25, 9.2 , 13.91,$

8.2, 13.45, 8.54, 13.34, 5.17, 7.7, 7.44, 8.74, 14.01,
5.53, 10.11, 6.39, 12.59, 6.03]

L = [8, 7, 9, 8, 7, 8, 8, 6, 9, 9, 9, 7, 9, 6, 7, 7, 8,
7, 9, 7, 9, 5, 6, 6, 8, 7, 7, 8, 6, 5, 7, 7, 10, 5,
6, 5, 6, 7, 5, 6, 8]

R = [0.72363754, 0.65135398, 0.79674574, 0.69468584, 0.74442395,
0.72748567, 0.61427651, 0.65477932, 0.6619443, 0.73156973,
0.63021967, 0.66195206, 0.69852031, 0.69098019, 0.65741823,
0.68618529, 0.75644166, 0.65312796, 0.75663052, 0.65274948,
0.72860079, 0.73703561, 0.7839319, 0.75942205, 0.72008043,
0.71230719, 0.7315298, 0.70147423, 0.57798889, 0.71766172,
0.71055233, 0.69250801, 0.6521953, 0.66585477, 0.61215604,
0.67304647, 0.73438835, 0.65560526, 0.66807371, 0.65957549,
0.7917005]

rho = [0.63, 0.55, 0.56, 0.63, 0.44, 0.17, 0.51, 0.83, 0.46, 0.59, 0.93,
0.79, 0.64, 0.57, 0.97, 0.68, 0.7, 0.26, 0.23, 0.29, 0.25, 0.19,
0.19, 0.66, 0.09, 0.8, 0.84, 0.83, 0.14, 0.46, 0.06, 0.91, 0.92,
0.56, 0.42, 0.37, 0.93, 0.87, 0.11, 0.29, 0.82]

The network has 2396 minimal paths from $s=1$ to $t=28$: They are written down in the MPs1.csv file.

We are interested in the estimation of the network reliability at level (36,120), i.e., we want to estimate the probability of sending 36 units of flow from the source s to the sink t within time 120 through one of 2396 minimal paths. The exact network reliability 0.8430803753527 has been determined using Kozyra's dictionary algorithm [1] and the serial bounding algorithm described in the article DOI: 10.1016/j.res.2021.107500. Based on the SSVs generated by the SSVGenerator agent and written by the TT agent down to the SSV1.csv file, the network reliability estimated by the TT agent is equal to **0.8423559199421269**. So, it differs from the exact reliability value by less than 0.01. If we repeated this process N times (for N tending to infinity), the number of cases where the approximated reliability differs from the exact one more than 0.01 would be approximately 0.01, since we have passed parameters $\epsilon = 0.01, \delta = 0.01$ to the SSVGenerator agent.

After activation agents in cmd you should obtain similar results and SSVs1.csv file:

```
java -cp JADE-all-4.6.0\JADE-bin-4.6.0\jade\lib\jade.jar;classes
jade.Boot -gui -agents
SSVAgent:SSVGenerator(0.01,0.01);TTAgent:TT(36,120)
```

```
lis 17, 2025 3:25:56 PM jade.core.Runtime beginContainer
```

```
INFO: -----
```

```
    This is JADE 4.6.0 - revision 6869 of 30-11-2022 14:47:03
    downloaded in Open Source, under LGPL restrictions,
    at http://jade.tilab.com/
```

```
-----
```

```
lis 17, 2025 3:25:56 PM jade.core.AgentContainerImpl
checkLocalHostAddress
```

```
WARNING:
```

```
*****
```

```
JAVA is not able to detect the local host address.
```

```
If this container is part of a distributed platform, use the
-local-host option to explicitly specify it
```

```
*****
```

```
lis 17, 2025 3:25:56 PM jade.imtp.leap.LEAPIMTPManager initialize
```

```
INFO: Listening for intra-platform commands on address:
```

```
- jicp://localhost:1099
```

```
lis 17, 2025 3:25:56 PM jade.core.BaseService init
```

```
INFO: Service jade.core.management.AgentManagement initialized
```

```
lis 17, 2025 3:25:56 PM jade.core.BaseService init
```

```
INFO: Service jade.core.messaging.Messaging initialized
```

```
lis 17, 2025 3:25:56 PM jade.core.BaseService init
```

```
INFO: Service jade.core.resource.ResourceManagement initialized
```

```
lis 17, 2025 3:25:56 PM jade.core.BaseService init
```

```
INFO: Service jade.core.mobility.AgentMobility initialized
lis 17, 2025 3:25:56 PM jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
lis 17, 2025 3:25:56 PM jade.mtp.http.HTTPServer <init>
INFO: HTTP-MTP Using XML parser
com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser
lis 17, 2025 3:25:56 PM jade.core.messaging.MessagingService boot
INFO: MTP addresses:
http://LAPTOP-T02T5KC7:7778/acc
lis 17, 2025 3:25:56 PM jade.core.AgentContainerImpl joinPlatform
INFO: -----
Agent container Main-Container@localhost is ready.
-----
Hallo! Transmission times computing-agent
TTAgent@localhost:1099/JADE is ready.

The aim is to estimate the probability of sending 36.0 units of flow
within time 120.0

Hallo! SSVGenerator-agent SSVAgent@localhost:1099/JADE is ready.

The minimum number of iterations is equal to 16588

It has been created the MFN with the following parameters:

W=[43,32,41,19,15,12,29,27,20,11,45,25,25,37,46,17,26,16,28,27,33,30
,47,23,31,10,20,28,31,36,12,44,26,41,38,16,21,28,35,37,42]

C=[6.25,14.65,9.55,13.95,12.0,6.5,6.89,9.36,13.72,7.9,6.66,8.14,13.8
4,10.53,7.28,8.53,8.92,10.22,5.39,7.09,12.2,11.32,10.15,7.7,11.25,9.
2,13.91,8.2,13.45,8.54,13.34,5.17,7.7,7.44,8.74,14.01,5.53,10.11,6.3
9,12.59,6.03]

L=[8,7,9,8,7,8,8,6,9,9,9,7,9,6,7,7,8,7,9,7,9,5,6,6,8,7,7,8,6,5,7,7,1
0,5,6,5,6,7,5,6,8]

R=[0.72363754,0.65135398,0.79674574,0.69468584,0.74442395,0.72748567
,0.61427651,0.65477932,0.6619443,0.73156973,0.63021967,0.66195206,0.
69852031,0.69098019,0.65741823,0.68618529,0.75644166,0.65312796,0.75
663052,0.65274948,0.72860079,0.73703561,0.7839319,0.75942205,0.72008
043,0.71230719,0.7315298,0.70147423,0.57798889,0.71766172,0.71055233
```

,0.69250801,0.6521953,0.66585477,0.61215604,0.67304647,0.73438835,0.65560526,0.66807371,0.65957549,0.7917005]

rho=[0.63,0.55,0.56,0.63,0.44,0.17,0.51,0.83,0.46,0.59,0.93,0.79,0.64,0.57,0.97,0.68,0.7,0.26,0.23,0.29,0.25,0.19,0.19,0.66,0.09,0.8,0.84,0.83,0.14,0.46,0.06,0.91,0.92,0.56,0.42,0.37,0.93,0.87,0.11,0.29,0.82]

16588 random SSVs have been generated!

Found the following transmission times computing agent:

TTAgent@localhost:1099/JADE

Estimated network reliability is equal to 0.8423559199421269

SSVGenerator-agent SSVAgent@localhost:1099/JADE terminating.