

Gsi – GERT simulation

Jiří Demel

This paper describes a simulation program Gsi for evaluation of GERT networks. This work was supported by research plan MSM6840770006. It was created on Department of Applied Informatics, Faculty of Civil Engineering, Czech Technical University in Prague.

1 GERT networks

GERT network is a model of a random proces. It has a form of directed graph with additional information.

Arcs of the graph represent activities. An activity is described by it's random duration and by a conditional probability that the activity is executed provided that its start node was activated. Several activities can be running in parallel.

Nodes of the graph represent events. There are several types of nodes. Each node is assigned type of input an type of output which together form type of a node.

Type of input can be either OR or AND.

OR Node of input type OR is activated by termination of any incoming activity.

AND Node of type AND is activated by termination of all incoming activities.

Type of output can be D (deterministic) or ST (stochastic).

D When deterministic node is activated all outgoing activities are started so that they run in parallel.

ST When stochastic node is activated exactly one of outgoing activities is randomly selected and started. For this purpose all outgoing arcs are assigned conditional probabilities that the arc will be started provided it's starting node is activated. Sum of those probabilities must be equal to one.

The description of input and ouput type of a node is combined to single word. So we have four types of nodes: STOR, STAND, DOR, DAND.

Additionally exactly one of nodes is defined to be start node. It is automatically activated at time 0.0.

In GERT networks with all nodes of type STOR at any model time only one activity can be running. This special sort of GERT networks can be evaluated by analytical methods.

General GERT networks are analyzed by the simulation based on the Monte-Carlo method, i.e. by performing random simulation experiments with the network an by statistical evaluation of the series experiments. In the sequel we call the experiment *simulation run*. Each simulation run starts at model time 0.0. The number of simulation runs is given by the `simulate` statement in input file.

2 Running the program

The program Gsi is written in Java, so Java Runtime Environment is necessary. The program can be run by command line

```
java --jar gsi.jar input_file [> output_file]
```

The program produces results on standard output. The standard output is usually redirected to a file.

Format of input file is described in the following section.

3 Input file

Input file is line oriented plain text file. Each line starts with a keyword and contains one or more parameters separated by spaces. All input lines except comments (//) and empty lines are copied or transformed to the output file.

The file starts with zero or more commands **model** and ends with command **simulate**. Order of all other lines is arbitrary. All keywords are case insensitive.

model may contain informal description of the GERT model. Each line of the description must start with the keyword *model*.

node *name node*

describes a node. The *name* is arbitrary text, the *type* is one of **stand**, **stor**, **dand**, **dor**.

arc *nodeFrom nodeTo probability distribution p1 p2 p3*

describes an activity where *nodeFrom* and *nodeTo* are nodes, *probability* is conditional probability of starting the activity if *nodeFrom* is activated. If *nodeFrom* is of deterministic type the probability is irrelevant. Distribution and its parameters are described later.

startNode *name*

defines the node that starts simulation.

stopAtTime *time*

defines a time at which the simulation run should stop.

seed *number*

defines seed of random number generator. If omitted a seed is derived from a system time of the computer. In any case the really used seed is listed in the output file.

logLevel *number*

is an integer between 0 and 5 which defines verbosity of the protocol of simulation.

simulate *number*

defines the number of simulation runs. Each simulation run starts at model time 0.0. This statement terminates the input file and starts processing of the model. Hence the rest of the input file is ignored.

// introduces a comment. The two slashes must be followed by a space. The rest of line is ignored. It can be used for temporary changes in the input file.

Random distribution of the duration is defined by a keyword and by one to three parameters.

keyword	p1	p2	p3	distribution
const	value	—	—	constant
exp	mean	—	—	exponential
uniform	min	max	—	uniform
gauss	mean	std. dev.	—	normal
triangle	min	max	mode	triangle
beta	min	max	mode	PERT-beta

Example input file:

```

model Example GERT network
model =====
model
model  (several lines of description)

node a1 DOR
arc  a1 a2  1.0  triangle  2.0  5.0  3.0
arc  a1 a3  1.0  const    2.0

node a2 dand
arc  a2 a4  1  uniform  2 3
arc  a2 a5  1  triangle 2. 5. 4.

node a3 STor
arc  a3 a4  0.3  exp  2.0
arc  a3 a5  0.7  exp  2.0

node a4 Dand
node a5 dand
arc  a5 a3  1.0  gauss  2 1

startNode  a1
stopAtTime 20.0

seed      1234567
logLevel  4
simulate  100

```

4 Simulation and output file

Output file consists of four parts.

4.1 Paraphrase of input file

First, the input file is paraphrased. It is not exact copy. If there are syntactic errors (e.g. missing values) they are reported and the program stops. So the last paraphrased line indicates position of the error. For example:

```

Node:  a2  dand
Arc: a2 --> a4  P=1.0  Distribution: uniform  2.0  3.0  0.0
Arc: a2 --> a5  P=1.0  Distribution: triangle  2.0  5.0  4.0
Node:  a3  stxor
Unknown type of node: stxor

```

4.2 Verification of the model

When statement `simulate` was reached in the input file all nodes and arcs are linked together and the structure of the resulting network is verified. If the verification was successful the output file contains

```
----- Model linked and verified -----
```

Otherwise errors are reported. For example:

```
Arc: a3 --> aX   P=0.7   Distribution: exp   2.0  0.0  0.0
Target node not found:  aX
```

4.3 Protocol of the simulation

Third part of the output file is a protocol of the simulation. It is useful for debugging of the model. Verbosity of the protocol depends on `logLevel` given in the input file.

- 0 no protocol at all
- 1 beginnings and ends of all single simulation runs
- 2 activations of nodes
- 3 started and stopped activities and their durations
- 4 tests whether a node can be activated (fired)
- 5 durations of activities.

Each simulation run is terminated either by overrunning of the model time given by parameter `stopAtTime`, or by a deadlock, i.e. the situation when all running activities stopped and nothing more can happen in the model. For example for `logLevel 2` the 95th simulation run may produce:

```
----- started simulation run 95 out of 100
0.0: node a1 fired
2.0: node a3 fired
3.8695678426750977: node a2 fired
7.095701339808068: node a5 fired
9.79219687986848: node a3 fired
10.716911183624642: Deadlock (empty queue) occurred
```

4.4 Statistics

The final part of the output file are statistics.

Note that random seed is reported even in the case when no random seed was given in the input file.

If there was no explicitly given random seed in the input file, the random number generator was initialized by a value that was derived from the system time of the computer. In such a case repeated runs of the program with identical input file would produce different output files. If an error or some anomaly appeared during the simulation, without knowledge of used random seed it would be impossible to repeat the same simulation with higher level of `logLevel`.

Next, for each node several values are reported. The *count* is the number of activations of the node in all simulation runs together. If *count* > 0 then *mean*, *stDev*, *min* and *max* are statistics of model times at which the node was activated. Note that during a simulation run a node can be activated more than once.

```
----- Statistics of 100 simulation runs -----
used random seed 1234567
```

```
Node  a1
count = 100
mean  = 0.0
stDev = 0.0
      min = 0.0
      max = 4.9E-324

Node  a2
count = 100
mean  = 3.5069466251177728
stDev = 0.6647532641162206
      min = 2.326633025019118
      max = 4.617772288789209

Node  a3
count = 163
mean  = 4.815648149803005
stDev = 3.677733573814523
      min = 2.0
      max = 13.50782628023614

Node  a4
count = 57
mean  = 7.915406664953615
stDev = 2.8918362371248945
      min = 4.436708311447584
      max = 15.3511677109553

Node  a5
count = 63
mean  = 7.355930576884992
stDev = 1.0375717043671417
      min = 5.068987839662446
      max = 12.384425613726554

----- End of Gsi -----
```