

SGNSim Ensemble Generator Manual

Jason Lloyd-Price and Andre S. Ribeiro

November 4, 2006

1 Introduction

To generate ensembles of gene regulatory networks as described by SGNSim, we provide a program that can create reaction systems based on a large set of parameters. Often many of these parameters will be fixed, and only a subset will vary depending on the goals of the user. Here we describe all available parameters, and how to manipulate them.

2 Running the Program

The program is generally executed from the command line as:

```
sgne -o output-filename -N N -t topology
```

Where *output-filename* is the filename of the simulator file produced, *N* is the number of genes in the network, and *topology* is the topology of the network (see section 3.1.1).

3 Input

Input to the generator is given in both the command line and a settings file. Network parameters such as the topology, and the transfer functions are given at the command line. The information needed to translate a network into reactions is given in a settings file.

3.1 Network Parameters

3.1.1 Topologies

The topology of the network is specified with the `-t` command line switch. All topologies require some extra parameters, such as the connectivity of the generated network. These parameters are given by the `-t:parameter-name` switch. Currently, there are two topologies integrated into the program: a randomly connected network with poisson input and output distributions, and a randomly connected network with a regular input distribution and a poisson output distribution. Both topologies have only one parameter: `k`, which is the mean connectivity of the network. For example, to generate a poisson-poisson network with $k = 2$, one would use:

```
sgne ... -t poisson-poisson -t:k 2 ...
```

It is also possible to load a topology from a file containing the directed connection information by specifying the `file` topology. The file is given by the parameter `file`. The file should be in one of two formats (given by the `filetype` parameter):

inputs: An input list where each line contains the output gene number, number of inputs, and finally the list of inputs. Gene numbers should be 0-based. For example, if gene 12 has inputs from genes 0, 4, and 7, then one would use:

```
12 3 0 4 7
```

iopairs: A list of input-output pairs, where each line contains the input gene number, and the output gene number. The example above would then be given by:

```
0 12
4 12
7 12
```

If **filetype** is not specified, the format is assumed to be **inputs**.

For example, to load a topology in the **iopairs** format from the file “**e. coli topology.dat**”, one would use:

```
sgne ... -t file -t:file "e. coli topology.dat" -t:filetype iopairs ...
```

This allows the user to use another program to generate any desired topology and easily input it into the generator.

3.1.2 Transfer Functions

Similar to topologies, transfer functions are specified with the **-b** command line switch, and parameters are given by the **-b:parameter**. Currently, only two transfer functions are available: **k** and **nn**.

The default transfer function is **k**. It will produce arbitrary functions with a bias towards ‘full activation’ given by the parameter **bias**. This bias is assumed to be 0.5 if left unspecified.

nn, on the other hand, will produce a subclass of linearly separable functions where the inputs are either activators or inhibitors, and never both. By default, a connection has an unbiased probability of being an activator. This can be changed by setting the **bias** parameter.

For example, to use **nn** functions with a **bias** of 0.7, one would use:

```
sgne ... -b nn -b:bias 0.7 ...
```

3.2 Settings File

The information required to translate a network topology and set of transfer functions into a full-fledged GRN simulation is stored in a settings file, which is **grnsim-params.txt** by default. The target file can be changed by setting the **-f** command line switch.

The layout of the settings file consists of two columns of strings separated by whitespace. The first column contains the setting names, and the second column contains their values. Comments can be added with **#**.

For example, to set the readout file of the simulation to **sim-results.out**, we would use:

```
readout-file          sim-results.out
```

3.2.1 Distributions

Many numeric settings which affect a large amount of elements in the simulation (such as genes and connections) allow a distribution to be given instead of a single number. Distribution names are listed here by their *root*. For example, **init-proteins** is the *root* for the distribution of the initial protein populations (see section 3.2.5). Several individual settings give the information required by a single distribution. The first of these is the distribution name, which is given by the setting **root-distr**. Each distribution takes several parameters, given by the **root-param#** settings. All parameters for a distribution must be specified. Available distributions are:

distr	param1	param2	Description
const, delta	d		Dirac delta spike at d
normal, gaussian	μ	σ	Normal distribution with mean μ and standard deviation σ
exponential, exp	λ		Exponential distribution with a mean of $1/\lambda$
uniform, minmax	min	max	A uniform distribution in the range $[min, max)$
poisson	λ		Poisson distribution with a given λ . Note that this distribution will only generate integer values.

A shortcut for the `const` distribution is available by only giving the `root` setting with the constant value. For integer fields, the sampled number will be floored.

For example, to set the distribution of the initial populations of proteins to a poisson distribution with $\lambda = 5$, we would use:

```
init-proteins-distr      poisson
init-proteins-param1    5
```

3.2.2 Simulation Settings

These settings affect the resulting simulation:

Setting	Type	Description
<code>simfile</code>	String	The output file containing the resulting reaction system
<code>run-time</code>	Real	The total time to run the simulation
<code>readout-interval</code>	Real	Sampling interval
<code>readout-file</code>	String	Output file of the simulation
<code>no-promoter-readout</code>	Boolean	If set to <code>true</code> , promoter states will not be output to the file

3.2.3 Gene Settings

These settings affect the properties of the genes:

Setting	Type	Description
<code>num-operating-sites</code>	Distribution	Number of operating sites per gene
<code>max-operating-sites</code>	Integer	Maximum number of operating sites per gene
<code>p-housekeeping</code>	Real	Probability of a gene having basal level synthesis
<code>no-opersite-competition</code>	Boolean	If set to <code>true</code> , all connections will be assigned a different operating site (and <code>num-operating-sites</code> is disregarded)
<code>no-inputs-is-housekeeping</code>	Boolean	If set to <code>true</code> , all genes with no inputs are given basal level, otherwise they are subject to <code>p-housekeeping</code>

3.2.4 Connection Settings

These settings affect the properties of individual connections among genes:

Setting	Type	Description
<code>p-heterodimer</code>	Real	Probability of merging a given pair of monomer inputs into a heterodimer connection
<code>p-homodimer</code>	Real	Probability of ‘upgrading’ a monomer connection to a homodimer connection (this is applied after <code>p-heterodimer</code>)
<code>p-indirect</code>	Real	Probability of making a connection indirect. There must always be at least one direct connection
<code>no-2-step</code>	Boolean	If set to <code>true</code> , transcription and translation will be merged into a single reaction
<code>no-normalize-rates</code>	Boolean	If set to <code>true</code> , division of the production reaction rates (transcription and translation) by the number of genes in the system will be disabled

3.2.5 Initial Concentrations

These settings affect the initial populations in the system:

Setting	Type	Description
<code>init-rbss</code>	Distribution	Number of the initial population of RBSs
<code>init-proteins</code>	Distribution	Number of the initial population of proteins
<code>init-dimers</code>	Distribution	Number of the initial population of dimers
<code>init-rnap-per-gene</code>	Integer	Initial number of RNAPs per gene
<code>init-rib-per-gene</code>	Integer	Initial number of Ribosomes per gene
<code>p-init-bound-promoter</code>	Real	Probability that a promoter's operating site will be initially bound to one of its inputs.

3.2.6 Reaction Rates

These settings describe distributions of rate constants for different reactions in the system. If a reaction has a rate of 0, it will be omitted.

Setting	Reactions Affected
<code>c-basal-level</code>	Basal level transcription reactions
<code>c-activated-level</code>	Activated level transcription reactions
<code>c-repressed-level</code>	Repressed level transcription reactions (usually 0)
<code>c-translation</code>	Translation reactions
<code>c-promoter-bind</code>	Promoter binding reactions
<code>c-promoter-unbind</code>	Promoter unbinding reactions
<code>c-indirect-unbind</code>	Indirect activation/inhibition reactions
<code>c-dimer-assoc</code>	Dimerization reactions
<code>c-dimer-disassoc</code>	Undimerization reactions
<code>c-rbs-decay</code>	RBS decay reactions
<code>c-protein-decay</code>	Protein decay reactions
<code>c-protein-decay-promoter</code>	Protein decay reactions where the protein is bound to a promoter's operating site

3.2.7 Reaction Delays

Reaction delays are one of the most important aspects of the simulation. Delays need to be set up in a slightly different manner than the rest of the settings since some delays are going to be highly dependent on some properties of the gene itself. For example, the time it takes for an RNA Polymerase to fully transcribe a gene depends greatly on the length of the gene. RNAPs also do not move at a constant velocity through the gene, and the delay ought to be drawn from a distribution as well [1]. To take both of these effects into consideration, SGNSim offers a way to set the run-time distribution of the time delays, as well as distributions for its parameters that are calculated immediately based on some global properties for the gene.

We will use the noisy RNAP delay as an example of how to set up a non-constant delay. It can be modelled as a normal distribution with a mean equal to $\mu = l/s + \tau_p$, where τ_p is the time to release the promoter, s is the translation speed, and l is the length of the gene in base pairs. The noise should scale with the length of the gene, so the standard deviation should follow $\sigma = dl$, where d is the standard deviation of the time it takes to transcribe a single base pair.

The gene length can be set in a parameter that can be used to modify the delays related to all genes, and is sampled from the distribution only once for each gene. There are 8 available, and can be used for any purpose. These global parameters are set by the distribution setting `d-param#`. Here, we will use `d-param1` as the gene length (l), and `d-param2` as the promoter release delay (τ_p). Then, to distribute gene lengths uniformly between 300 and 3000 base pairs, we would use:

```

d-param1-distr      uniform
d-param1-param1    300
d-param1-param2    3000

```

The promoter release time can also be generated from a distribution.

The delay we want to set is the `d-rnap` delay (see table below). To set the run-time distribution of the time delay, use the `root-distr` setting (see the simulator manual for the list of run-time distributions). In this case, we want to use a `normal` distribution. This program does not know anything

about the runtime distributions, so the number of parameters must be set by the `root-nparams` setting. The `normal` distribution takes 2 parameters, so to set up the `normal` run-time distribution, we would use:

```
d-rnap-distr          normal
d-rnap-nparams       2
```

We must now calculate the mean time it takes for the RNAP to be released: $\mu = l/s + \tau_p$. Each parameter of the run-time distribution is generated first from a source distribution (set by the setting `root-param#-source`), and is then modified by a series of elementary operations involving the global parameters. We will use the source distribution for the first parameter to set s to 200 with the following settings:

```
d-rnap-param1-source-distr  const
d-rnap-param1-source-param1 200
```

First, we divide the gene length (stored in global parameter 1) by this value. To accomplish this, there are several is a list of elementary operations that can be use to modify the source variable. These are set in the settings `root-param#-op#`, and are read in the format “*operation [global parameter index]*”. Available operations are:

Operation	Effect
<code>nop</code>	Nothing
<code>add</code>	Add the global parameter to the current number
<code>mult</code>	Multiply the global parameter to the current number
<code>div</code>	Divide the global parameter to the current number
<code>min</code>	Take the <i>min</i> of the global parameter and the current number
<code>max</code>	Take the <i>max</i> of the global parameter and the current number

The global parameter index can be any of the 1-8 parameters defined above, or 0 to reference the source distribution for the delay.

In this case, we want to have $\mu = l/s + \tau_p$, where s is stored in global parameter 1 and τ_p is stored in global parameter 2, so we would set:

```
d-rnap-param1-op0      add 1
d-rnap-param1-op1      div 0
d-rnap-param1-op2      add 2
```

There are 9 available ops, with indices in the range 0-8. By default, `op0` is `add 0`, and all others are `nops`. This system does not allow a completely arbitrary function of the parameters to be built, but provides much more flexibility than would otherwise be available.

We can build a similar construction for the parameter $\sigma = dl$, where d is 0.001:

```
d-rnap-param2-source-distr  const
d-rnap-param2-source-param1 0.001
d-rnap-param2-op1          mult 1
```

Available delay setting *roots* are:

Setting	Delays Affected
<code>d-promoter</code>	Promoter release delay after transcription
<code>d-rnap</code>	RNAP release delay after transcription
<code>d-rbs</code>	RBS release delay after transcription
<code>d-rbstrans</code>	RBS release delay after translation
<code>d-ribosome</code>	Ribosome release delay after translation
<code>d-protein</code>	Protein release delay after translation

When `no-2-step` is set, the transcription+translation reaction has delays drawn from the `d-promoter`, `d-rnap` and `d-protein`, where the protein delay should account for the delays from transcription and translation.

3.2.8 Defaults

Most settings must be specified. Default values for those who are not required are listed here:

Setting	Default
init-rnap-per-gene	10
init-rib-per-gene	10
run-time	100
readout-interval	1
readout-file	genenet-output.xls
p-init-bound-promoter	0.0
no-inputs-is-housekeeping	true
no-2-step	false
no-opersite-competition	false
no-promoter-readout	false
no-normalize-rates	false
max-operating-sites	10
d-param#	Constant distribution with $d = 0$

A Complete Example

Suppose we want to generate ensembles of networks with 10 genes of average connectivity 2. We would run the program with the command line:

```
sgne -N 10 -t:k 2
```

The program will then read all other parameters in the settings file (`grnsim-params.txt`):

```
##### Simulation Settings #####
simfile          sgn.sim # Simulation file
run-time         10000 # Total simulation time
readout-interval 10 # Simulation sampling interval
readout-file     sgn-results.xls # Output file
no-promoter-readout true # Disable promoter state readout

##### Initial Concentrations #####
init-rbss-distr  uniform
init-rbss-param1 1 # Minimum initial number of RBS's
init-rbss-param2 50 # Maximum initial number of RBS's
init-proteins-distr uniform
init-proteins-param1 0 # Minimum initial number of proteins
init-proteins-param2 0 # Maximum initial number of proteins
init-dimers-distr uniform
init-dimers-param1 0 # Minimum initial number of dimers
init-dimers-param2 0 # Maximum initial number of dimers
init-rnap-per-gene 10 # Initial amount of RNAP per gene
init-rib-per-gene 10 # Initial amount of Ribosomes per gene
p-init-bound-promoter 0 # Chance that a gene's promoter will be
                        # initially bound to one of its input
                        # proteins

##### Gene Settings #####
num-operating-sites-distr const
num-operating-sites-param1 3 # Number of operating sites per gene
max-operating-sites 15 # Maximum number of operating sites
p-housekeeping 0.1 # Chance that a gene is a housekeeping gene
```

```

# (has basic level)
no-operosite-competition      false # Remove operating site competition -
                                # will ignore num-operating-sites and
                                # number them sequentially

#####      Connection Settings      #####
p-heterodimer                 0.0 # Chance that a pair of inputs to a gene will
                                # be merged into a heterodimer connection
p-homodimer                   0.0 # Chance that a connection will be 'upgraded'
                                # to a homodimer connection
p-activator                   0.5 # Chance that any connection is an activating
                                # connection
p-indirect                    0.0 # Chance that a connection will be an
                                # indirect one
no-inputs-is-housekeeping     true  # Set to 'true' to automatically make any
                                # gene with no inputs a housekeeping gene
no-2-step                     false # Set to 'true' to have basic/activated
                                # level reactions immediately produce
                                # proteins
no-normalize-rates           false # Set to 'true' to disable normalization
                                # of the production rates by the number
                                # of genes

#####      Reaction Rates      #####
c-basal-level-distr           uniform
c-basal-level-param1          1 # Basal level rate min
c-basal-level-param2          1 # Basal level rate max
c-activated-level-distr       uniform
c-activated-level-param1      1 # Activated level rate min
c-activated-level-param2      1 # Activated level rate max
c-repressed-level-distr       const
c-repressed-level-param1      0 # Repressed level rate
c-translation-distr           uniform
c-translation-param1          1 # Translation rate min
c-translation-param2          1 # Translation rate max
c-promoter-bind-distr         uniform
c-promoter-bind-param1        1 # Promoter binding rate min
c-promoter-bind-param2        1 # Promoter binding rate max
c-promoter-unbind-distr       uniform
c-promoter-unbind-param1      1 # Promoter unbinding rate min
c-promoter-unbind-param2      1 # Promoter unbinding rate max
c-indirect-unbind-distr       uniform
c-indirect-unbind-param1      1 # Indirect unbinding rate min
c-indirect-unbind-param2      1 # Indirect unbinding rate max
c-dimer-assoc-distr           uniform
c-dimer-assoc-param1          1 # Dimer association rate min
c-dimer-assoc-param2          1 # Dimer association rate max
c-dimer-disassoc-distr        uniform
c-dimer-disassoc-param1       1 # Dimer disassociation rate min
c-dimer-disassoc-param2       1 # Dimer disassociation rate max
c-rbs-decay-distr             uniform
c-rbs-decay-param1            0.01 # RBS decay rate min
c-rbs-decay-param2            0.01 # RBS decay rate max
c-protein-decay-distr         uniform
c-protein-decay-param1        0.01 # Protein decay rate min
c-protein-decay-param2        0.01 # Protein decay rate max

```

```

c-protein-decay-promoter-distr    uniform
c-protein-decay-promoter-param1  0.01    # Protein decay rate min while bound
                                     # to a promoter
c-protein-decay-promoter-param2  0.01    # Protein decay rate max while bound
                                     # to a promoter

#####      Reaction Delays      #####
d-param1-distr                    uniform
d-param1-param1                  360 # Min gene length
d-param1-param2                  360 # Max gene length
d-param2-distr                   const
d-param2-param1                  2   # Promoter/RBS release delay
d-promoter-distr                 const
d-promoter-nparams               1
d-promoter-param1-source-distr   const
d-promoter-param1-source-param1  0
d-promoter-param1-op1            add 2
d-rnap-distr                     const
d-rnap-nparams                   1
d-rnap-param1-source-distr       const
d-rnap-param1-source-param1      0.05 # Average time for RNAP to process
                                     # a single base pair

d-rnap-param1-op1                mult 1
d-rnap-param1-op2                add 2
d-rbs-distr                      const
d-rbs-nparams                    1
d-rbs-param1-source-distr        const
d-rbs-param1-source-param1       0   # Amount of time between the promoter release
                                     # and the RBS release (can be negative)

d-rbs-param1-op1                 add 2
d-rbstrans-distr                 const
d-rbstrans-nparams               1
d-rbstrans-param1-source-distr   const
d-rbstrans-param1-source-param1  0   # RBS release delay when translating RBS to proteins
d-rbstrans-param1-op2            add 2 # Add the promoter delay to it
d-ribosome-distr                 const
d-ribosome-nparams               1
d-ribosome-param1-source-distr   const
d-ribosome-param1-source-param1  0.05 # Average time for a Ribosome to process
                                     # a single base pair

d-ribosome-param1-op1            mult 1
d-ribosome-param1-op2            add 2
d-protein-distr                  const
d-protein-nparams                1
d-protein-param1-source-distr     const
d-protein-param1-source-param1    0.05 # Average time for a Ribosome to process
                                     # a single base pair

d-protein-param1-op1              mult 1
d-protein-param1-op2              add 2

```

References

- [1] Nicholas A. M. Monk. Oscillatory expression of *hes1*, *p53*, and *nfκb* driven by transcriptional time delays. *Current Biology*, 13:1409–1413, 2003.