CANTATA user manual

Summary

The program can be called with the following options:

-
  ```
  cantata --optimize -n <network file> -r <rule file>
  [-o <result file>] [-on <output network files>]
  [-me <max error>] [-ps <population size>]
  [-no <number of offspring>] [-if <frac. injected original networks>]
  [-nf <allow negation every xth offspring>]
  [-ni <number of iterations>] [-ns <number of restarts>]
  [-ms <max states>] [-mt <max transitions>]
  [-tw <topology fitness weight vector>]
  [-im <initial mutations for new individuals>]
  [-eps <epsilon>] [-rs <random seed>] [-q]
  ```

-
  ```
  cantata --validate -n <network file> -r <rule file>
  [-ms <max states>] [-mt <max transitions>] [-c <collapse>]
  [-ds] [-rs <random seed>] [-q]
  ```

Here, the first option is the main switch specifying the function that is started. Other parameters are either key-value combinations (where `<...>` is a placeholder for the value) or switches. Square brackets denote optional parameters.

Main switches

The first command line argument is interpreted as the main switch, which determines function that is started. Available main switches are:

`--optimize`	Starts the *CANTATA* algorithm for evolutionary reconstruction of Boolean networks.
`--validate`	Checks a model for violations of rules that specify the expected dynamics.

The functions in detail

`--optimize`

Starts the *CANTATA* algorithm for evolutionary reconstruction of Boolean networks. The algorithm requires a network model draft in parameter `-n`. It transforms the model to match the expectation specified specified in the rule file in parameter `-r` and writes candidate models to the standard output or an output file whose name can be defined in the optional parameter `-o`.
The parameters in detail:

`-n`	The name of the file that contains the input network model draft. The model must be specified in the *BoolNet* network file format which is described in Section “File formats”.
`-r`	The name of the rule file that describes the expected dynamics of the network. The rule file format is described in Section “File formats”.
`-o`	An optional parameter specifying the name of the result file. This file contains a list of reconstructed candidate network models with their fitness values. By default (if this parameter is not specified), the results are printed to the standard output.
`-on`	An optional parameter specifying a file name pattern for separate candidate network model files. If this parameter is set, a *BoolNet* network file is created for each candidate having a maximum error (i.e., a maximum value of the first fitness function accumulating the violations of dynamic expectations) of `-me`. The file name should be a pattern that contains a marker `%d` which is replaced by a running number for the candidate networks.
`-me`	The maximum value of the first objective value (i.e. the accumulated rule violations) for which a separate network model file should be created. Defaults to 0, which means that files are only created for networks that match the rules perfectly.
`-ps`	The population size, i.e. the number of individuals that survive each generation of the evolutionary algorithm. Defaults to 100.
`-no`	The number of offspring that are created in each generation of the evolutionary algorithm. The offspring consist of a fraction of mutated individuals from the last generation and a fraction of mutated copies of the input network, depending on the `-if` parameter. Defaults to 200.
`-if`	The fraction of offspring that are created as mutated copies of the input network instead of mutated copies of the previous generation. Defaults to 0.1.
`-nf`	The frequency of applications of the negation mutation operator. The algorithm is allowed to apply the negation operator to at most every `-nf`-th offspring.
`-ni`	The number of iterations (or generations) for the evolutionary algorithm. Defaults to 1000.
`-ns`	The number of times the evolutionary algorithm is restarted. Defaults to 1.

`-ms`	The maximum number of states that are tested to calculate the violations of the rule set and the robustness of the network. The larger this number is, the more accurate is the calculation, but at the cost of a longer runtime. Defaults to 200.
`-mt`	The maximum number of state transitions that are calculated for each tested start state to reach an attractor or chain. Defaults to 100.
`-tw`	An optional vector of weights for the subscores in the topology fitness function. This must consist of three floating point numbers, separated by slash characters. Here, the first number specifies the weight for the size term, the second number specifies the weight for the similarity term, and the third number specifies the weight for the penalty term for constant genes. The weights must sum up to 1. The default is `0.25/0.25/0.5`.
`-im`	The number of mutations applied to the input model draft to create a new individual. Defaults to 1.
`-eps`	The ϵ used in the comparison of objective scores. If the difference in the scores of two individuals is less than ϵ, they are considered as equal. Defaults to 0.0005.
`-rs`	The seed of the random number generator. This seed can be fixed in order to obtain reproducible results. Defaults to the current time stamp in milliseconds.
`-q`	If this option is set, the algorithm will run quietly without printing any status and info messages except the results.

`--validate`

Compares the dynamics of the input network model `-n` with the expected dynamics specified in parameter `-r` using the Dynamic Programming matching. For attractor rules, the matchings and the corresponding rule violations are printed. Furthermore, a list of start states is printed for each of the matchings. For sequence/chain rules, a list of violations and the start states causing these violations are printed. As each start state yields a separate matching for chain rules, the actual matchings are not printed here for the sake of clarity. The mismatches and the corresponding start states are printed to the standard output or an optional output file that can be specified in the parameter `-o`.
The parameters in detail:

`-n`	The name of the file that contains the input network model. The network must be specified in the *BoolNet* network file format which is described in Section .
`-r`	The name of the rule file that describes the desired dynamics of the network. The rule file format is described in Section “File formats”.
`-o`	An optional parameter specifying the name of the result file. This file contains a list of violations of the supplied rules caused by the supplied network model. By default, the results are printed to the standard output.
`-ms`	The maximum number of states that are tested to calculate the violations of the rule set. The larger this number is, the more accurate is the calculation, but at the cost of a longer runtime. As this is a one-time check, a high default value of 100 000 is used.
`-mt`	The maximum number of state transitions that are calculated for each tested start state to reach an attractor or chain. Defaults to 1000.
`-c`	The maximum number of start states that are printed for each violation. If more states violate the corresponding specification, only the first states are printed. Defaults to 50.
`-ds`	A switch specifying that start states should be drawn deterministically. This means that the first start state to be used is the state where all free genes are 0, and the next states are generated by adding one to the decimal representation of the free genes. If this flag is not set and `-ms` is smaller than the maximum number of possible states, states are sampled at random.
`-rs`	The seed of the random number generator. This seed can be fixed in order to obtain reproducible results. Defaults to the current time stamp in milliseconds.
`-q`	If this option is set, the algorithm will run quietly without printing any status and info messages except the results.

File formats

BoolNet network file format

This section provides a full language description for the network file format. The network file starts with a header line

```
targets, factors
```

Each of the following lines describes the Boolean transition function of one target gene. The line consists of the name of the target gene, a comma, and the Boolean expression for the transition function. e.g.

```
gene1, (gene2 | !gene3) & gene4
```

Here, `|` denotes the logical OR, `&` denotes the logical AND, and `!` denotes a negation. Optionally, a certainty value in form of a probability can be supplied (which is an extension to the original format in *BoolNet*). This certainty value influences the probability of changing the corresponding function in the optimization process. E.g.,

```
gene1, (gene2 | !gene3) & gene4 [1.0]
```

specifies that we are completely certain about this transition function and do not want the algorithm to change it. Certainty values only occur in input networks – the candidate networks returned by the algorithm do not contain them.
The following details the format in Extended Backus-Naur Form (EBNF).

```
Network           = Header Newline
                    {Function Newline};
Header            = "targets" Separator "factors";
Function          = GeneName Separator BooleanExpression [  "[" Probability "]" ];
BooleanExpression = GeneName
                    | "!" BooleanExpression
                    | "(" BooleanExpression ")"
                    | BooleanExpression " & " BooleanExpression
                    | BooleanExpression " | " BooleanExpression;
GeneName          = ? A gene name from the list of involved genes ?;
Separator         = ", ";
Probability       = ? A floating-point number ?;
Newline           = ? A line break character ?;
```

Rule file specification

A rule file can be subdivided into rule block, each specifying a single rule. A rule can either describe an attractor (attractor rule) or a sequence of states corresponding to a time series (chain rule). An attractor rule starts with the keyword `Attractor:`, a chain rule with `Chain:`. This header is followed by several sections:

- `Initial condition:` This section consists of a line specifying the condition under which the rule applies. The condition specifies the values certain genes must take before the corresponding chain or attractor is reached. Initial states are generated according to these gene values, while unspecified genes are set to random values. The section is required. However, the gene value list may be intentionally left empty, which means that the rule applies to any start state.

- `Fixed genes:` This optional section consists of a line specifying which genes are knocked out or overexpressed when the rule is applied. This means that these genes are constantly fixed to 0 (knock-out) or 1 (overexpression) independent of the corresponding transition functions of the network when the rule is applied. By contrast, the initial condition only specifies the initial state of a gene, but does not influence its further behaviour. The initial condition and the list of fixed genes must be compatible , i.e. a gene cannot take an initial value that differs from the setting in the list of fixed genes. In general, the list of fixed genes is a subset of the genes specified in the initial condition. Similarly, state specification entries may not contradict the list of fixed genes.

- `State specifications:` This section holds the state specification list. Each of the following lines is a specification entry specifying the gene values for one state of the attractor or time series. If there are several alternatives, the first alternatives starts with `State specifications:`, and all further alternatives start with `Or:`. At least one alternative is required.

- `Importance:` This section contains an optional importance weight (a floating-point number) for the rule. That is, the scores of the rules are weighted according to the importance values in the fitness function. By default, all rules are weighted equally with 1.

Initial conditions, fixed genes and state specifications are supplied as conjunctions. Here, a conjunction is specified as a list of literals (i.e., genes or negated genes), separated by whitespaces.
Lines starting with `#` are considered as comments and ignored.

The following example shows a typical rule file with one attractor rule for a two-state attractor and a chain rule for a three-state sequence:

```
Attractor:
Initial condition:
Gene1 Gene2
State specifications:
Gene1 Gene2 Gene3
Gene1 !Gene2 Gene3

Chain:
Initial condition:
!Gene1 Gene2
State specifications:
!Gene1 Gene2 !Gene3
!Gene1 !Gene2 !Gene3
!Gene1 Gene2 Gene3
```

Here, the attractor is only reached if both Gene 1 and Gene 2 were initially active, i.e. from states with `Gene1=1` and `Gene2=1`, whereas the time series is only reached if Gene 1 was initially inactive and Gene 2 was initially active (`Gene1=0` and `Gene2=1`).
The following example specifies the expected response to a knock-out experiment:

```
Attractor:
Initial condition:
!Gene1
Fixed genes:
!Gene1
State specifications:
Gene2 !Gene3
```

That is, we expect to reach a steady state with active Gene 2 (`Gene2=1`) and inactive Gene 3 (`Gene3=0`) if we knock out Gene 1 (i.e. `Gene1` is always 0).
If there are several plausible alternative attractors that can be reached under a certain condition, we can define alternative state specification lists:

```
Attractor:
Initial condition:
Gene1 Gene2
State specifications:
# first alternative: two-state attractor
Gene1 Gene2 Gene3
Gene1 !Gene2 Gene3
Or:
# second alternative: steady-state attractor
Gene1 !Gene2 !Gene3
Importance:
2
```

In this case, two attractors can be reached from start states with `Gene1=1` and `Gene2=1`: Either the two-state attractor, or a steady-state attractor. In addition, this rule has an increased importance weight, which means that the rule influences the rating of a network twice as much as usual.
The following provides the complete file format specification in EBNF:

```
Rule                = Header
                      InitialCondition
                      [FixedGenes]
                      StateSpecifications
                      {Alternative}
                      [Importance];
Header              = ("Attractor:" | "Chain:") Newline;
InitialCondition    = "Initial condition:" Newline
                      Conjunction;
FixedGenes          = "Fixed genes:" Newline
                      Conjunction;
StateSpecifications = "State specifications:" Newline
                      Conjunction Newline
                      {Conjunction Newline};
Alternative         = "Or:" Newline
                      Conjunction Newline
                      {Conjunction Newline};
Importance          = "Importance:" Newline
                      FloatingPointNumber;
Conjunction         = {[!] GeneName " "};
GeneName            = ? The name of a gene also present in the network draft ?;
FloatingPointNumber = ? A floating-point number ?;
Newline             = ? A line break character ?;
```