ASSA-PBN 3.0: Analysing Context-sensitive Probabilistic Boolean Networks
(Accompanying Tool Tutorial)

This documentation is part of the tool artefact files submitted to the conference CMSB 2018. It is used for demonstrating the new functionalities of version 3.0 of ASSA-PBN. A complete user guide of the tool is available at the tool website `http://satoss.uni.lu/software/ASSA-PBN/`.

1 System Requirements

1. Platform: x86 compatible 32 bit or 64 bit processor;
2. Operating system: Windows, Linux and Mac OSX (10.4 and above);
3. Required environment: Java SE Development Kit 8 or higher;
4. Required compilers (for attractor detection): flex 2.5.4 or higher, GNU bison 2.3 or higher, GNU g++ 4.0.1 or higher; Cygwin 1.5.25-14 or higher (Windows platform only).

2 Installation

ASSA-PBN can be obtained at the tool website `http://satoss.uni.lu/software/ASSA-PBN/#download`. For basic usage, ASSA-PBN does not require installation operation. For usage in Windows, the software can be run by double clip on the assa.jar file after extracting the downloaded file. For usage in Mac OS of Linux, the software can be run with the command "./xassa" in the "bin" folder. Alternatively, the command line version can be run with the command "./assa <parameters>" in the "bin" folder.

Installation for the attractor detection function. If the user wants to use the attractor detection function, an additional installation is required. It can be down by the following two steps.

1. (Windows platform only) Install cygwin and the packages g++, flex and bison on Windows. Detailed instructions can be found from `http://www.cygwin.com/`.

2. In a terminal (please use the cygwin terminal for Windows), navigate to the mcmas folder. Simply use the command make to the compile MCMAS. We now give an example in Windows. Assume the downloaded ASSA-PBN is extracted in the folder "C:\assa-pbn-3.0.1". Open the cygwin terminal, run "cd C:\assa-pbn-3.0.1" and then run "make". On Windows, the directory bin in the cygwin installation must be added to the system path. For example, append c:\cygwin64\bin to the system path if cygwin is installed in c:\cygwin64.

Installation for computation with GPUs. If the user wants to benefit from the parallel computation power of GPU codes, the NVIDIA's CUDA Compiler (NVCC) (`https://developer.nvidia.com/cuda-downloads`) are required. The parallel computation of GPU is currently only supported for Mac OS and Linux. In addition, an additional installation step (see below) is required in order to enable the GPU-based parallel computation.

– In a terminal, navigate to the ASSA-PBN folder and run `make`.

ASSA-PBN can be run either in command line or in a GUI.

3 Defining a CPBN

In the newly released version, ASSA-PBN provides a high-level PBN definition format to support the definition of a CPBN. We will use an example CPBN with 3 nodes to demonstrate how to define a CPBN using the high-level format.

context	functions	probability
0	$f_1 = x_0 \& x_1$ $f_2 = (!x_2) \& (x_1 \mid x_0)$ $f_3 = (!x_0) \& x_1$	0.1
1	$f_1 = x_0 \& x_1$ $f_2 = (!x_2) \& (x_1 \mid x_0)$ $f_3 = x_0 \& x_1 \& (!x_2)$	0.2
2	$f_1 = x_0 \& x_1$ $f_2 = x_0 \& x_1 \& (!x_2)$ $f_3 = (!x_0) \& x_1$	0.7

Table 1: Boolean functions and their selection probabilities of a 3-node CPBN.

The example CPBN contains three nodes x_0, x_1, and x_2 and three contexts. The Boolean functions of the three contexts and their selection probabilities are shown in Table 1. In addition, this CPBN has a perturbation rate of 0.001 and a switch probability 0.01.

The original high-level CPBN definition file for this model can be found in the "examples" folder of the downloaded software package, with the name 'example-context-sensitive-highlevel.pbn'. We also show it in Figure 1. The description contains four parts. The first part is composed of the first three lines, specifying the model type, the number of nodes and the perturbation rate respectively. The second part is between the keywords `nodeNames` and `endNodeNames`. It lists the names of all nodes one by one. The order of the names determines the order of the nodes in ASSA-PBN. In ASSA-PBN, each node is assigned a unique non-negative integer as the ID of the node. Therefore, the order in fact determines the ID of the nodes in ASSA-PBN. A node name cannot start with a number and cannot contain the following characters: space, tab, $*, \&, !, |, \hat{}, /, :, (,)$. The third part specifies the distribution for all the contexts, the probability to make a switch and the Boolean functions details for all the nodes. The specification of Boolean functions for a node starts with the keyword `node`, which is followed by the node name, and ends with the keyword `endNode`. Each line between the two keywords specifies one Boolean function and its contexts in the format of 'context indices : function'. For example, the last line on the left of Figure 1 specifies one Boolean function for node x_0 and this function is shared by contexts 0,1, and 2. The functions can be defined using parenthesis and the four different logical operations, i.e., logical

type=synchronous	**endNode**
n=3	**node** x_1
perturbation=0.001	0,1 : $(!x_2)\&(x_1\|x_0)$
nodeNames	2 : $(!x_0)\&x_1\&x_2$
x_0	**endNode**
x_1	**node** x_2
x_2	0,2 : $(!x_0)\&x_1$
endNodeNames	1 : $x_0\&x_1\&(!x_2)$
contextDistribution=0.1, 0.2, 0.7	**endNode**
switch=0.01	**npNode**
node x_0	x_0
0,1,2 : $x_0\&x_1$	**endNpNode**

Fig. 1: An example of context-sensitive PBN high level definition file.

and &; logical or |; logical not !, and logical exclusive or ^. Note that the logical not operation is required to be surrounded by a pair of parenthesis, e.g., $(!x_2)$. The last part specifies the nodes that should not be perturbed. It starts with the keyword `npNode` and ends with `endNpNode`. If all the nodes should be perturbed, this part can be omitted.

4 Starting the GUI of ASSA-PBN

Open a terminal and change the directory to the "bin" folder of ASSA-PBN, and run the command "./xassa". The main interface of ASSA-PBN will be launched. The interface is divided into three parts: the menu at the top, three panels in the middle and the status bar at the bottom. The panels are used to display PBN information and running information. See Figure 2 for an example. The panels shown in this figure contain the information of a loaded model and steady-state computation results. We will demonstrate how this information is obtained in the following examples.

5 Loading a PBN Model into ASSA-PBN

Launch the main interface of ASSA-PBN after installation. Select the menu option 'Model | Load', then select the file, e.g., 'example-context-sensitive-highlevel.pbn' and click 'Open'. This example PBN is then loaded into ASSA-PBN as shown in Figure 2.

The general information of the loaded PBN, e.g., the number of nodes, the network density, the updating mode, the perturbation rate, and the function details, is shown in the top left panel. The function details are shown as a tree structure in the panel. When a predictor function is selected, the function details will be shown in the bottom panel as a truth table. The top right panel is used to show all the running information.

6 Computing Steady-state Probabilities

One major functionality offered by ASSA-PBN is the computation of steady-state probabilities of a PBN. We demonstrate how to use the two-state Markov chain approach to

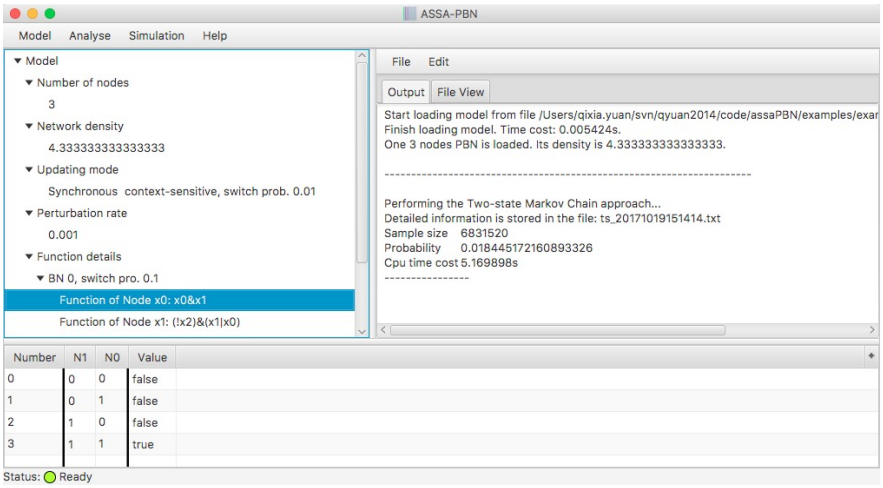
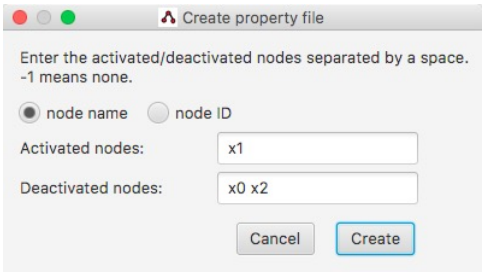


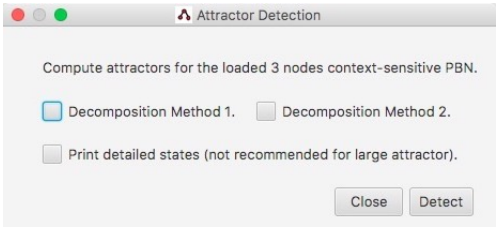
Fig. 2: Screenshot of the main interface.



(a) Launch of the two-state MC approach



(b) Creating a property



(c) Launch the attractor detection function.

Fig. 3: Screenshots of three pop-up windows.

compute steady-state probabilities of the example network used in Section 5. Select the menu option 'Analyse | Steady State Probability | Statistical | Two State Markov Chain'. A window, shown in Figure 3a (on page 4), pops up. This pop-up window asks for six parameters. The 'precision' and 'confidence level' are used to define how precise the computed result should be. We set precision and confidence level to 10^{-3} and 0.95, respectively. The 'GlobalAlias' and the structure parallel are only meaningful for IPBNs

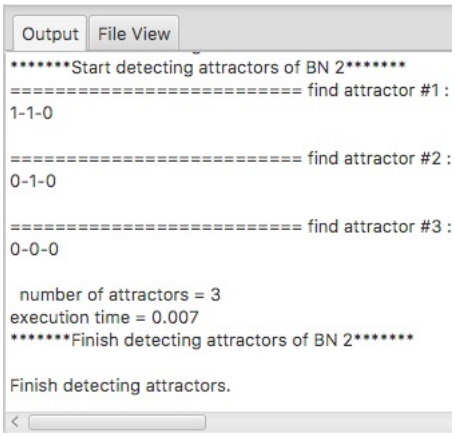


Fig. 4: Attractor detection result.

and hence they are disabled for this loaded context-sensitive PBN. 'Properties' defines the set of states whose steady-state probability is to be computed. It can be assigned either by clicking the three dots button, which will lead to a file selection window; or by clicking the 'Set properties' button, which will lead to a window for assigning activated and deactivated nodes as shown in Figure 3b. This figure defines the set of states where node x_1 is active and nodes x_0 and x_2 are inactive. We click the 'Create' button and save the property in a file named 'property_context.txt'. The two (exclusive the gray ones) radio selections are used to specify how the simulation should be performed. The two gray ones are valid for IPBNs. It can be either in a sequential way, or a parallel way. The parallel ways use the CPU-based parallel technique to gain more speed. If the CPU-based parallel is selected, the gray text field box '# cores' will become active and it will be filled with the available number of CPU-cores in the current machine. This value determines the number of chains used in the multiple CPU-based parallel two-state Markov chain approach. Users can change this value; however, the simulation speed will not increase if the value is larger than the actual number of CPU cores. In this demonstration, we use the sequential way. The main information, i.e., the extending times, the sample size, the probability, and the time cost of the computation, is shown in the top right panel of ASSA-PBN's main interface, as shown in Figure 2. More detailed information about this run can be obtained by clicking the 'File View' tab. Demonstrations for computing steady-state probabilities with other methods can be found in the user guide.

7 Attractor Detection

We demonstrate in this section one major contribution of the new version ASSA-PBN, i.e., attractor detection. After loading the model as mentioned in the Section 5, select the menu option 'Analyse | Attractor Detection'. A window will pop up asking for choices to perform the task. See Figure 3c for demonstration. ASSA-PBN provides three meth-

ods for attractor detection. One non-decomposition method (the default one) and two decomposition methods. If the user do not tick any of the first two choice boxes, the default non-decomposition method will be used. Since ASSA-PBN supports both synchronous and asynchronous updating schemes, and the detection methods for different updating schemes are different, there are in fact six different methods implemented.

We refer to [1] for the details of the decomposition method. In general, for large networks, we recommend the decomposition method 2 for efficiency. In this example, the loaded model is a 3-node synchronous network. Hence, we use the default one, i.e., without ticking any of the first two choice boxes. The detected result will be shown in the top right panel, as in Figure 4. It shows the results of the last BN (context) of the loaded CPBN. Three singleton attractors are detected.

8 Parameter Estimation

When performing parameter estimation of a CPBN, we are interested in estimating the switching probabilities of the contexts of the PBN. To reach this goal, we first perform experiments in the lab and measure the steady-state probabilities of interested nodes in different conditions, e.g., wild type and mutants. Each condition corresponds to one PBN. We then use parameter estimation techniques to find the optimal parameters which make the PBNs fit the lab experimental data best. To perform parameter estimation in ASSA-PBN, we need to specify the different PBN models under different conditions, and the lab experimental data. Those information are required to be stored in specification files. We use the specification files in the folder 'examples' of the downloaded package to demonstrate parameter estimation.

To perform parameter estimation, select menu option 'Analyse | Parameter Estimation' to launch the parameter estimation window as shown in Figure 5. Click 'Add PBN files' button to add files 'example-context-sensitive-highlevel.pbn' and 'example-context-sensitive-highlevel_2.pbn' in the 'examples' folder. Then click the second three dots button and select the file 'context-sensitive-highlevel_properties.txt' to assign experimental data. We do not assign nodes to be fitted. This option is only valid for IPBNs. The format of experimental data file is explained in details in the user guide. The parameter estimation method drop down list provides the available parameter estimation methods. Currently only particle swarm method is available. The following four parameters are related to the particle swarm method. We set them as shown in Figure 5. If 'Allow parallel evaluation?' is checked, the particle swarm method will be run in parallel, which means that in each iteration, n particles will be evaluated in parallel, where n is determined by the number of cores. We refer to the user guide for the remaining 5 parameters. After setting all the parameters and clicking the 'Compute' button, the parameter estimation is launched and the result is shown in the top-right panel. The result includes detailed information of each iteration and the final estimated parameter values. A screenshot of the result is shown in Figure 6.

References

1. Mizera, A., Pang, J., Qu, H., Yuan, Q.: Taming asynchrony for attractor detection in large Boolean networks (techinical report). (2017) https://arxiv.org/abs/1704.06530.

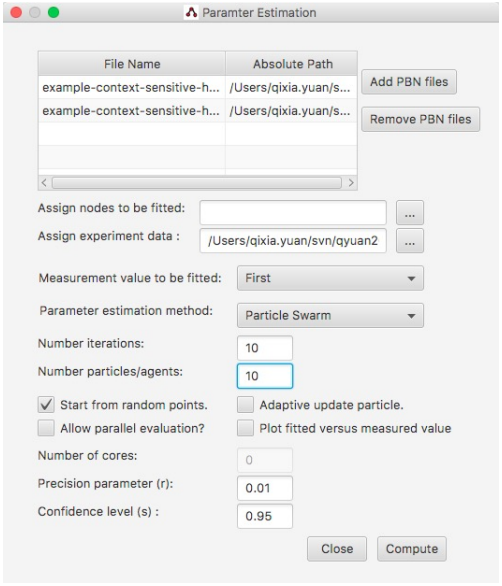


Fig. 5: Interface of parameter estimation.

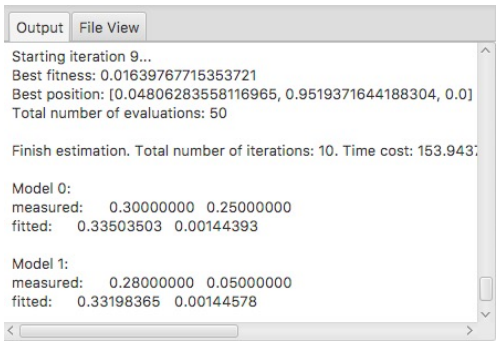


Fig. 6: Screenshot of parameter estimation output.