StBound Tool

Approach combining stochastic bounds and histograms for performance analysis of queue

F. Aït-salaht[1], H. Castel-Taleb[2],J.M. Fourneau[3] and N. Pekergin[4]

[1] LIP6, Université Paris Ouest, Nanterre, France
{farah.ait-salaht}@lip6.fr
[2] SAMOVAR, UMR 5157, Télécom Sud Paris, Evry, France
hind.castel@telecom-sudparis.eu
[3] DAVID, Université de Versailles-Saint-Quentin, France
jean-michel.fourneau@uvsq.fr
[4] LACL, Université Paris Est, France
nihal.pekergin@u-pec.fr

1 Installation

The ***StBound*** tool is composed by two applications "***BHa*** and ***ASingle_Queue***" implemented with Matlab 2012a. They were subsequently compiled, using the Matlab compiler, into a program that can be run outside of Matlab environment (in the form of Windows executables (.exe)).

Our applications are therefore available in two forms : an open-source Matlab application (Matlab license required) and a stand-alone Java executable (free).

1.1 Requirements

With MATLAB

- Verify that the MATLAB Compiler Runtime (MCR) is installed
- If the MCR is not installed, run MCRInstaller.exe provided in archived files "***BHa.zip***" or "***ASingle_Queue.zip***"

Without Matlab

- In order to run these executable programs, an appropriate version of the MCRInstaller (we provide Windows version) has to be downloaded from archived file "***BHa.zip***" or "***ASingle_Queue.zip***"
- User can also download MCR Matlab Compiler Runtime for free from the Mathworks website
- Installation of the MCR (for Windows) : Double-click on the *MCRInstaller.exe* and follow the instructions

Once the MCR is installed on a user machine, the MCR Installer never needs to be ran again.

For more information on the MCR Installer, see the MATLAB Compiler documentation.

1.2 Installation

- Download the ***StBound.zip*** archive.
- Unzip it somewhere on your computer.

1.3 Getting started

- Left-click on Program_pkg.exe. This will unpack the files.
- Left-click on Program.exe to run. Note that it may take around 30 sec to load when running it for the first time.

2 BHa application : Bounding Histogram approach

"Real traffic traces, histograms and stochastic bounding approach"

A picture of the main window is given below :

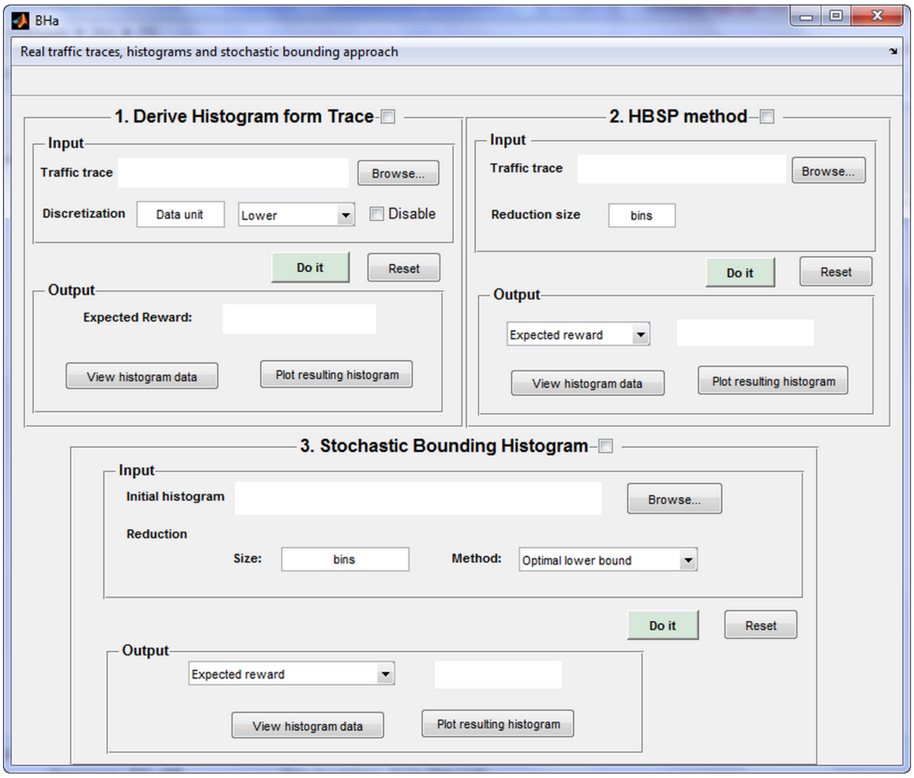


FIG. 1 – The main window BHa application

This application includes three programs :

1. **Derive Histogram from Trace :** construct the histogram (discrete distribution) corresponding to the input trace.

2. **HBSP method :** derive HBSP histogram method developed by Hernández-Orallo [1].

3. **Stochastic Bounding Histogram :** for an input histogram defined on N states, this program allows to compute stochastic bounding histogram defined on reduced size $K << N$ using the following methods : Optimal [2], Greedy[2] or Tancrez [3] approaches.

We detail below these different components.

2.1 First program : Derive histogram from trace

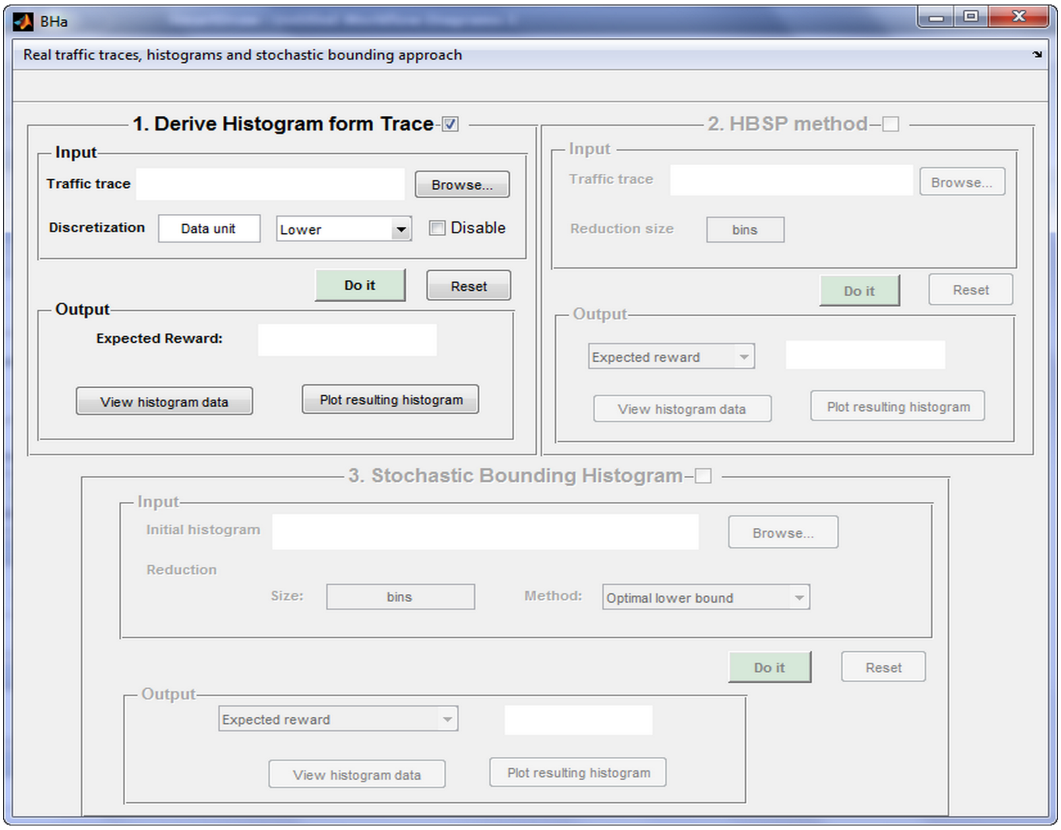
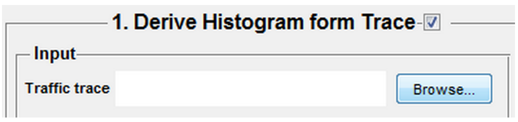


FIG. 2 – Derive histogram from trace program.

2.1.1 Input parameters

1. **Trace data file**. Insert the trace file.



This file must respect the following format :

```
% NameTraceFile.txt

        number of state in the trace
        number of bits in the first T period
        number of bits in the second T period
        ....
        ...
```

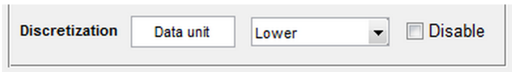
`Example.` Example of trace defined on five frames and saved in *data.txt* file.

```
% data.txt

5
4
9
```

```
1
12
6
```

2. **Discretization.** If the user wants apply a discretization on the input trace he should set the data unit value and select the kind of bound he want to employ for his resolution, otherwise he checks the box **Disable**.



2.1.2 Output parameters

By pushing **Do it** button, the program returns the following results :

- Expected reward of a histogram corresponding to the input trace
- A file containing the histogram corresponding to the input traffic trace. Depending on the type of discretization used, the following file will be created :
  - **Hist_trace.txt**, if the user do not use discretisation and button Disable is checked
  - **Hist_trace_L**, if we apply discretisation on lower bound
  - **Hist_trace_U**, if we apply discretisation on upper bound

  These files have the following format :

  ```
  % Hist_trace.txt

        number of state in the histogram
        state1      probability of state1
        state2      probability of state2
        ...
        ...
        ...
  ```
- The program also allows to view the obtained histogram by clicking on **View histogram data** button.

One input example (**T_Mawi2007F_40ms1h.txt**) is included in "Input_Output" folder for testing purposes.

2.2 Second program : HBSP method

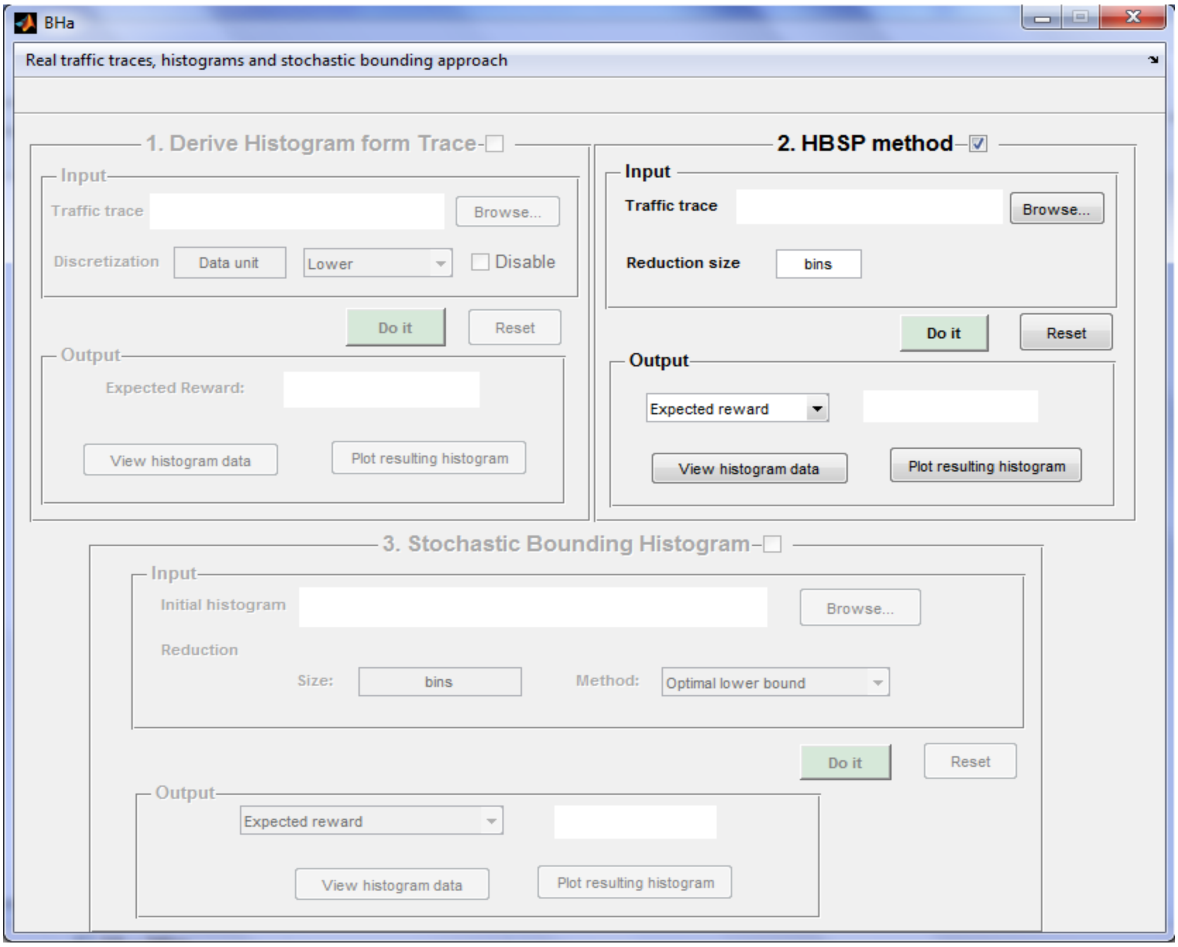


FIG. 3 – HBSP method program.

2.2.1 Input parameters

- Trace data file
- Size of reduction : *bins*

2.2.2 Output parameters

By pushing **Do it** button, the program returns the following results :

- The expected reward of the resulting HBSP histogram
- The execution time in second (*i.e.* the time required to compute the HBSP histogram)
- A file named "HBSP_hist.txt" containing the HBSP histogram. This file will be created in "Input_Output" folder
- The program also allows to view the obtained histogram by clicking on *View histogram data* button and plot the histogram by clicking on *Plot resulting histogram* button

2.2.3 Example

Considering **T_Mawi2007F_40ms1h.txt** file (included in "Input_Output" folder) containing the MAWI traffic trace [4], the execution of the second program allows us to obtain the results illustrated below.

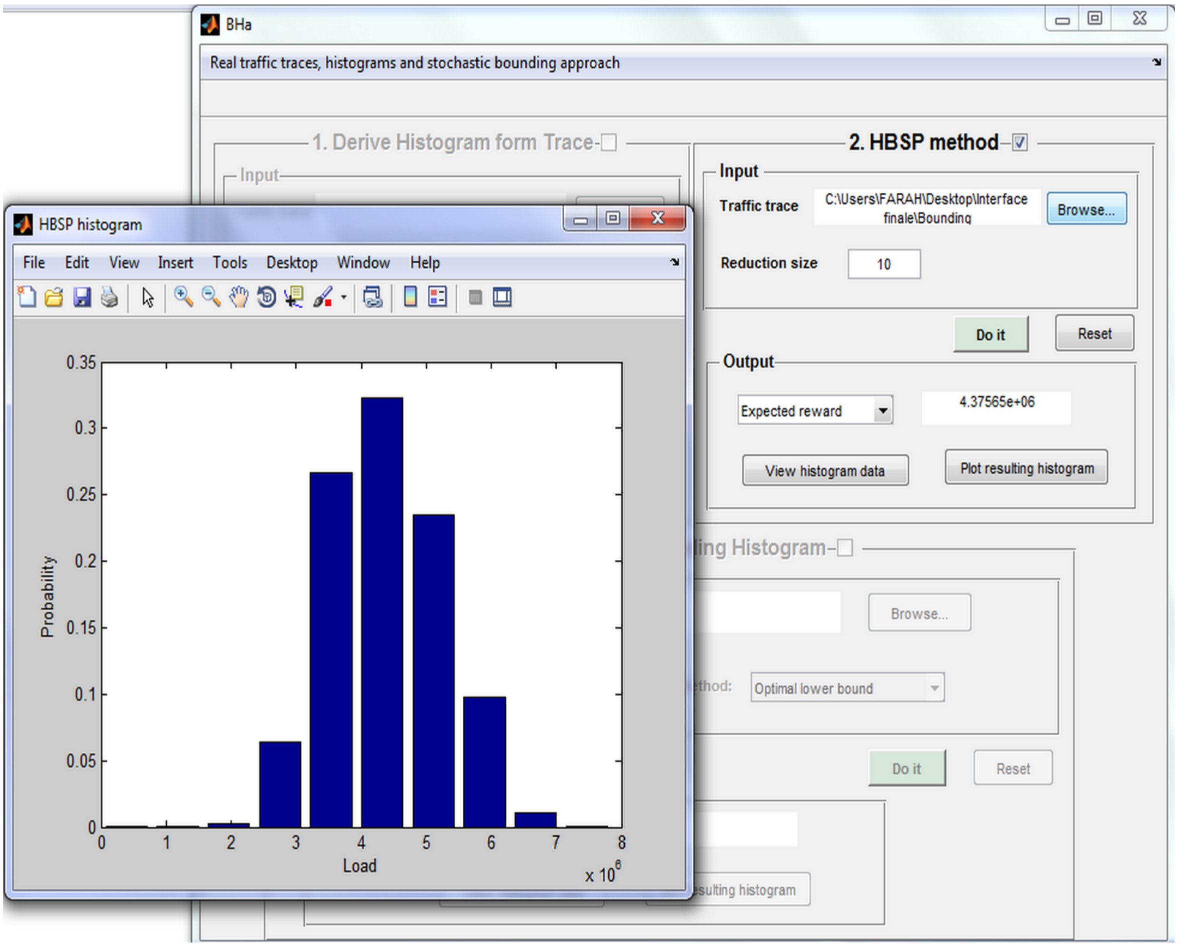


FIG. 4 – HBSP method program.

2.3 Third program : Stochastic Bounding Histogram

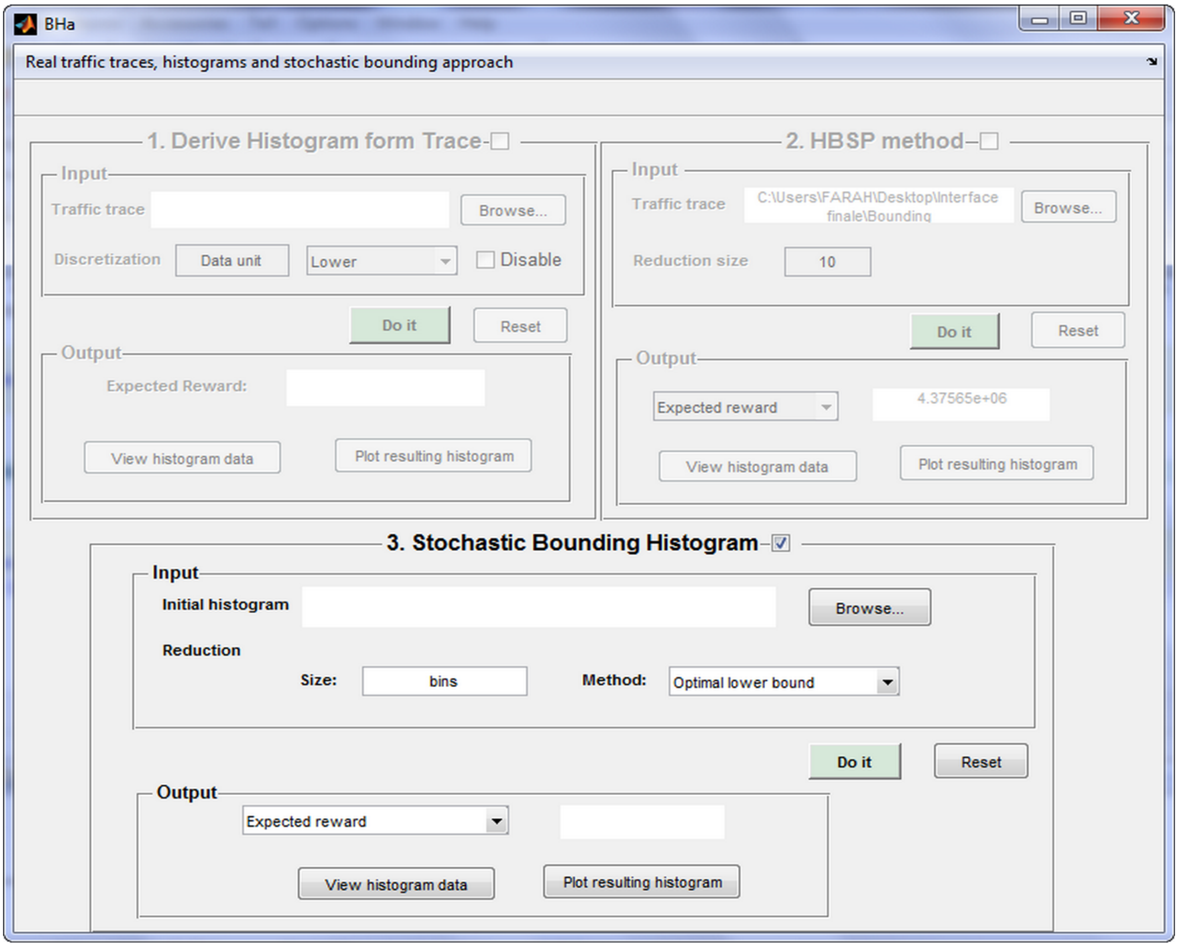


FIG. 5 – Stochastic Bounding Histogram program.

2.3.1 Input parameters

- Initial histogram file
- Reduction size and reduction method (Optimal lower bound, Optimal upper bound, Greedy lower bound, Greedy upper bound and Tancrez upper bound)

2.3.2 Output parameters

By pushing **Do it** button, the program returns the following results :

- The expected reward of the bounding histogram
- The execution time in second (*i.e.* the time required to compute the resulting histogram)
- A file containing the bounding histogram. Depending on the type of the method we employed, the following file will be created :
  - **OptLowerBoundHist.txt**, for the Optimal lower bound
  - **OptUpperBoundHist.txt**, for the Optimal upper bound
  - **GreedyLowerBoundHist.txt**, for the Greedy lower bound
  - **GreedyUpperBoundHist.txt**, for the Greedy upper bound
  - **TancrezUpperBoundHist.txt**, for the Tancrez upper bound

  We note that these files will be created in "Input_Output" folder.

– The program also allows to view the obtained histogram by clicking on ***View histogram data*** button and plot the histogram by clicking on ***Plot resuting histogram***button

One input example (**Hist_trace_L.txt**) is included in "Input_Output" folder for testing this program valid for the lower bound method.

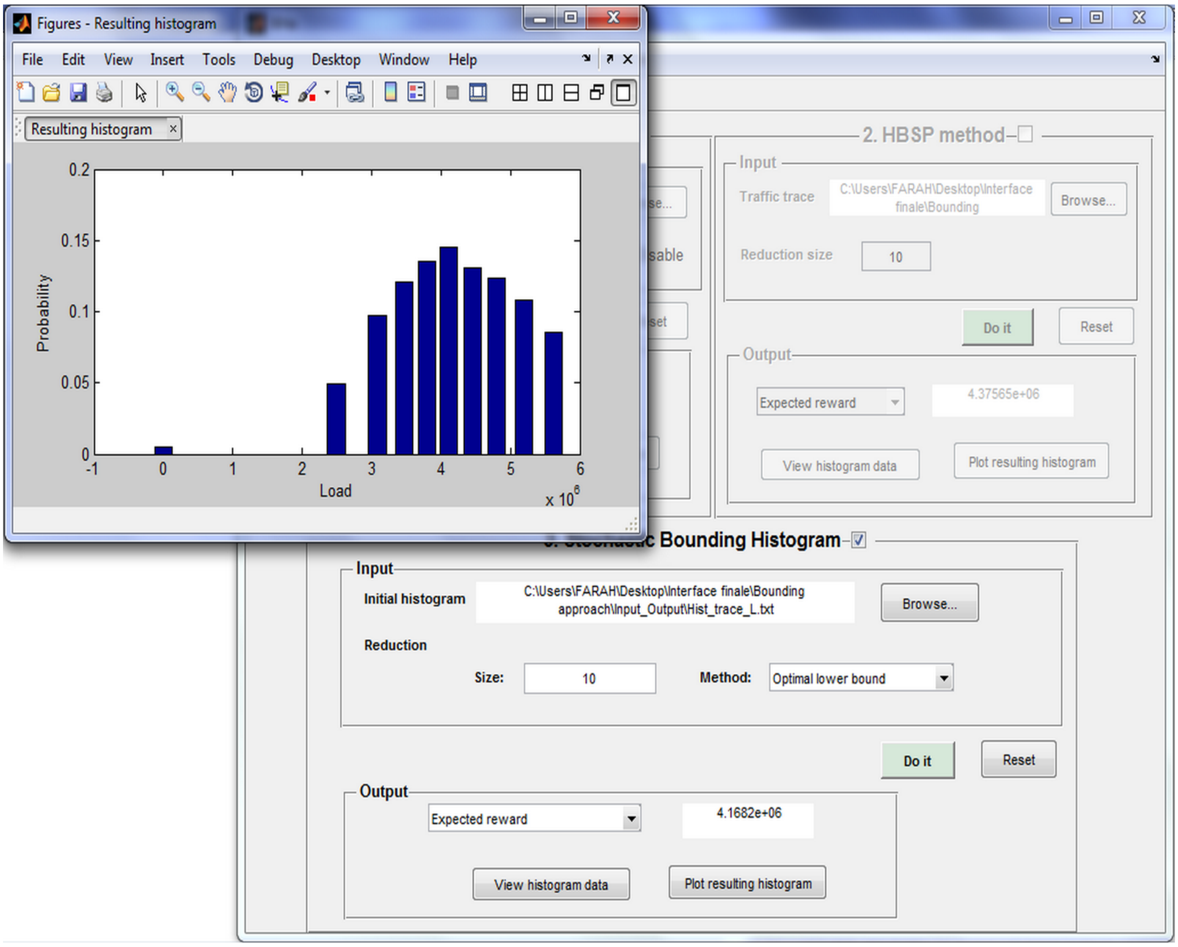


FIG. 6 – Example.

3 Analysis of single queue

In order to analyze the model of a single queue illustrated in Figure 7, we developped an application called **ASingle_Queue** presented in Figure 8.

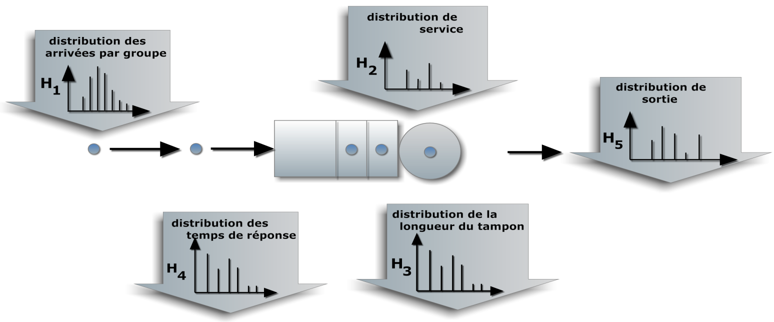


FIG. 7 – Input and output parameters of a queueing model.

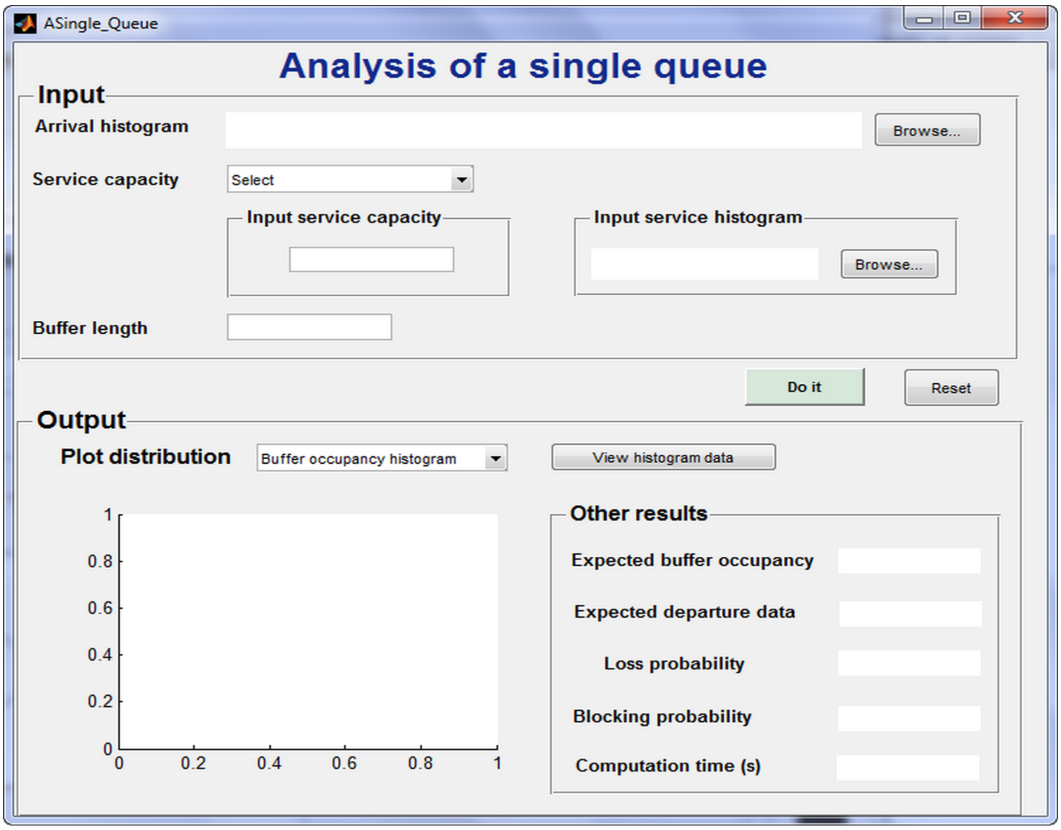


FIG. 8 – ASingle_Queue application.

3.0.3 Input parameters

- Arrival histogram file
- Kind of service :
  - **Deterministic**. In that case, the user should introduce the service capacity
  - **Variable**, *i.e.* discrete distribution (histogram). Here, the user should introduce the service histogram file
- Buffer length

3.0.4 Output parameters

By pushing **Do it** button, the program returns the following results :
- The expected buffer occupancy
- Expected departure data
- Loss probability
- Blocking probability
- The computation time in second (*ie.* the time required to compute all performance measures of the queue)
- Creation of the following files :
  - **BufferOccupancyHist.txt**, contains the Buffer occupancy histogram
  - **DepartureHist.txt**, contains Departure histogram
  - **LossesHist.txt**, contains Losses histogram
- View the different cumulative distribution function of the output histograms (illustrated on the graphic)
- The program also allows to view the obtained histogram (couple : [state, probability]) by clicking on ***View histogram data*** button

3.1 Example

We consider a single queue with arrival histogram corresponding to the optimal lower bound with number of bins equal to 20. The service is assumed deterministic with capacity equal to 4.4×10^6 bits (*ie.* 110 Mbs with sampling period T=40ms) and the buffer length is set to 10^6 bits.

The use of **ASingle_Queue** application allows us to evaluate the performance of an isolated queue as illustrate in the following figure.

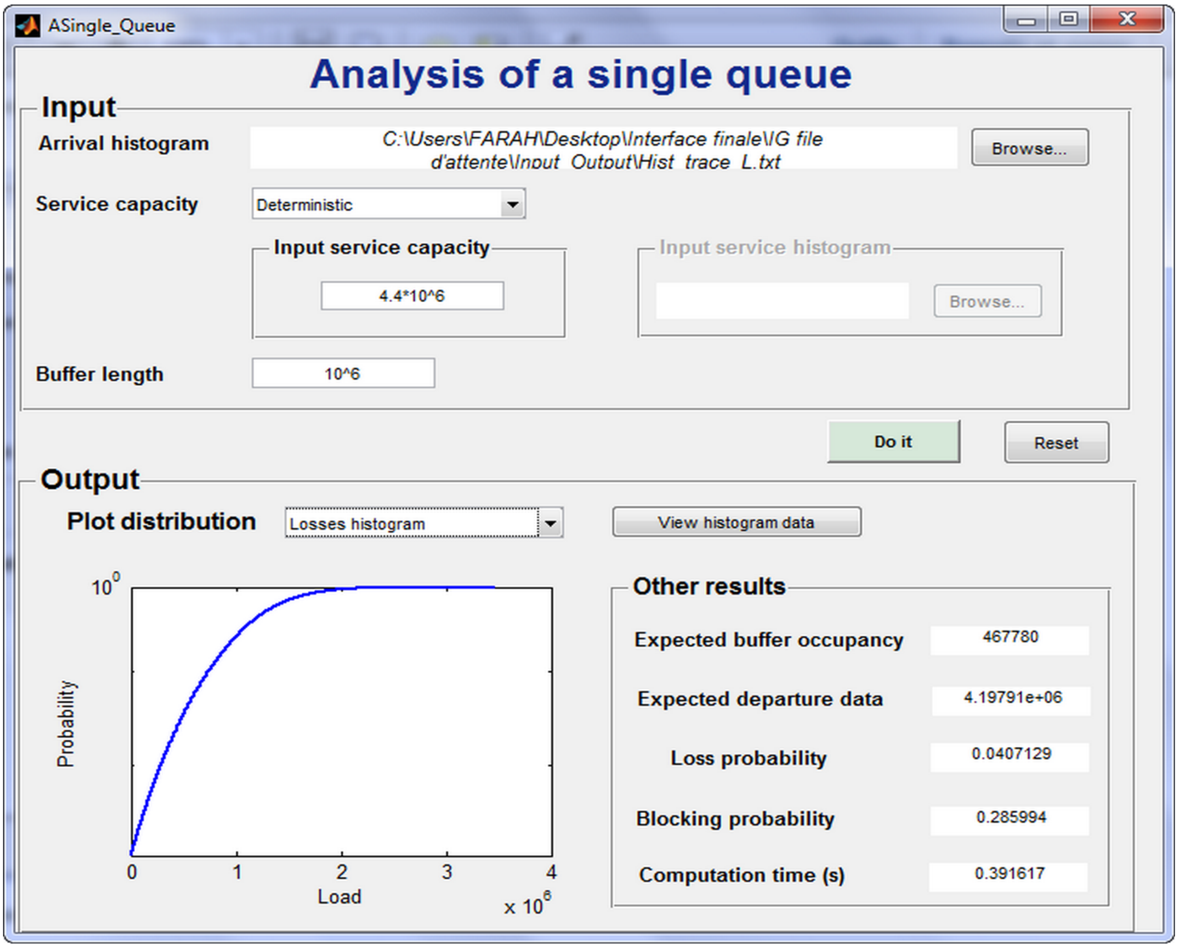


FIG. 9 – Analysis of single queue.

4 References

[1] E. Hernández-Orallo and J. Vila-Carbó. A histogram-based stochastic process for finite buffer occupancy analysis. In VALUETOOLS, page 44, 2007.

[2] F. Aït-Salaht, J. Cohen, H. Castel Taleb, J. M. Fourneau, and N. Pekergin. Accuracy vs. complexity : the stochastic bound approach. In 11th International Workshop on Disrete Event Systems (WODES 2012), number 8, pages 343-348, 2012.

[3] J. S. Tancrez, P. Semal, and P. Chevalier. Histogram based bounds and approximations for production lines. European Journal of Operational Research, 197(3) : 1133–1141, 2009.

[4] Kenjiro Cho Sony and Kenjiro Cho. Traffic data repository at the wide project. In In Proceedings of USENIX 2000 Annual Technical Conference : FREENIX Track, pages 263–270, 2000.

[5] F. Aït-Salaht, H. Castel Taleb, J. M. Fourneau, et N. Pekergin. Stochastic bounds and histograms for network performance analysis. In *10th European Workshop on Performance Engineering (EPEW'13)*, volume 8168, pages 13–27, 2013.

[6] F. Aït-Salaht, H. Castel Taleb, J. M. Fourneau, et N. Pekergin. A bounding histogram approach for network performance analysis. In *15th IEEE International Conference on High Performance Computing and Communications (HPCC'13)*, China, 2013.