Performance evaluation and applications project presentation

Project A

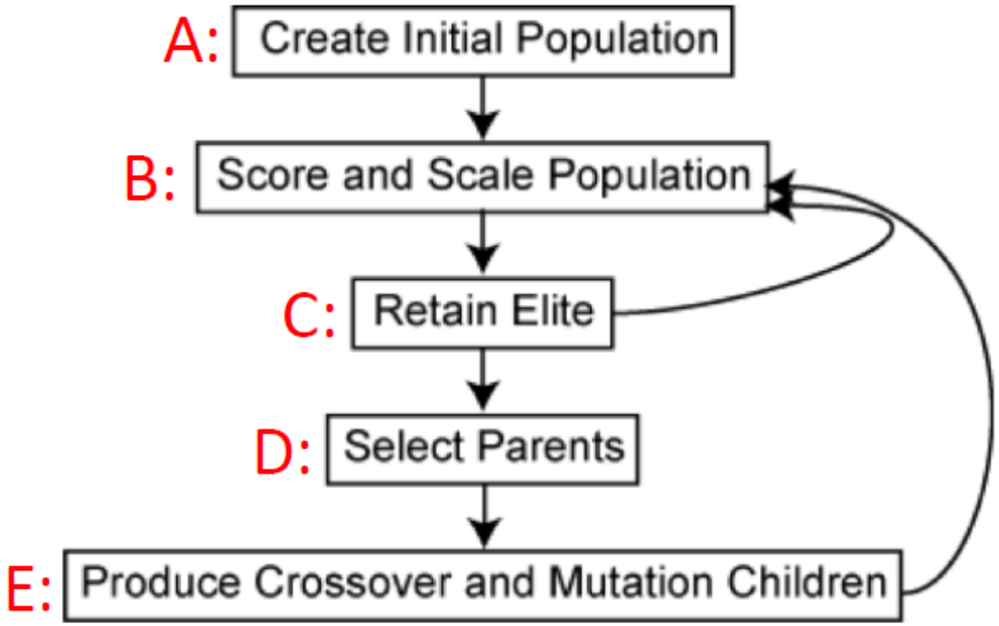
Prof. Marco Gribaudo

Student: Samuel Kala
10584699

Problem description

A genetic algorithm is composed by five stages, and they are executed according to the figure.
The probability of repeating stage B or continue to stage D after stage C, and the one of returning to stage B or finish the algorithm after stage E, have been computed:

- pCtoB = 0.9
- pEtoB = 0.8



Problem description

The algorithm is continuously repeated: as soon as solution has been found, it is run again on a new problem. The **runtime of execution** of the stages (in milliseconds) is **collected in trace files**: TraceA-A.txt, TraceA-B.txt, TraceA-C.txt, TraceA-D.txt, TraceA-E.txt If the algorithm is run on a multi-core machine, stages B and E can be fully parallelized: the **run time will be perfectly divided by the number of cores**. The other stages would not obtain any benefit from parallelization, and would run on a single CPU, taking exactly the same time.

- The system administrator would like to know the **minimum number of cores** required to provide an average of more than **30 solutions per second**. Which would be the **average utilization of the CPU** in this case?

Approach To The Solution

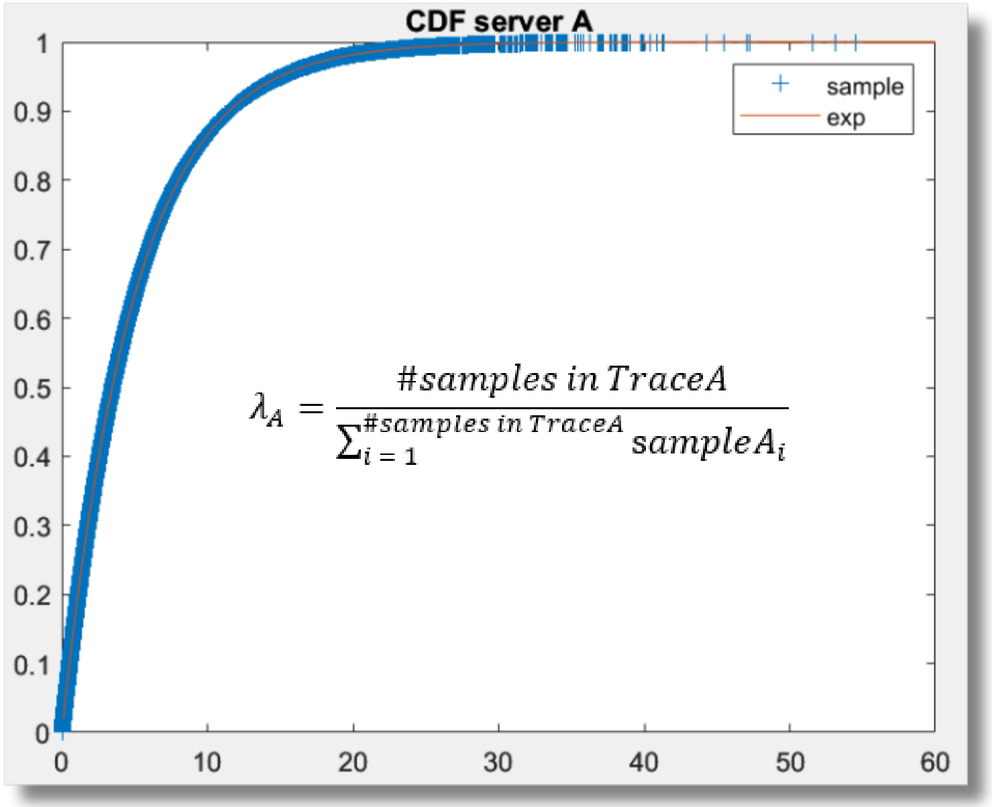
The phases taken to arrive to the final solution were the following:

1. **Fitting** of the data provided in the Trace files
2. **Decision** of the model
3. **Computation** of the required performance indexes

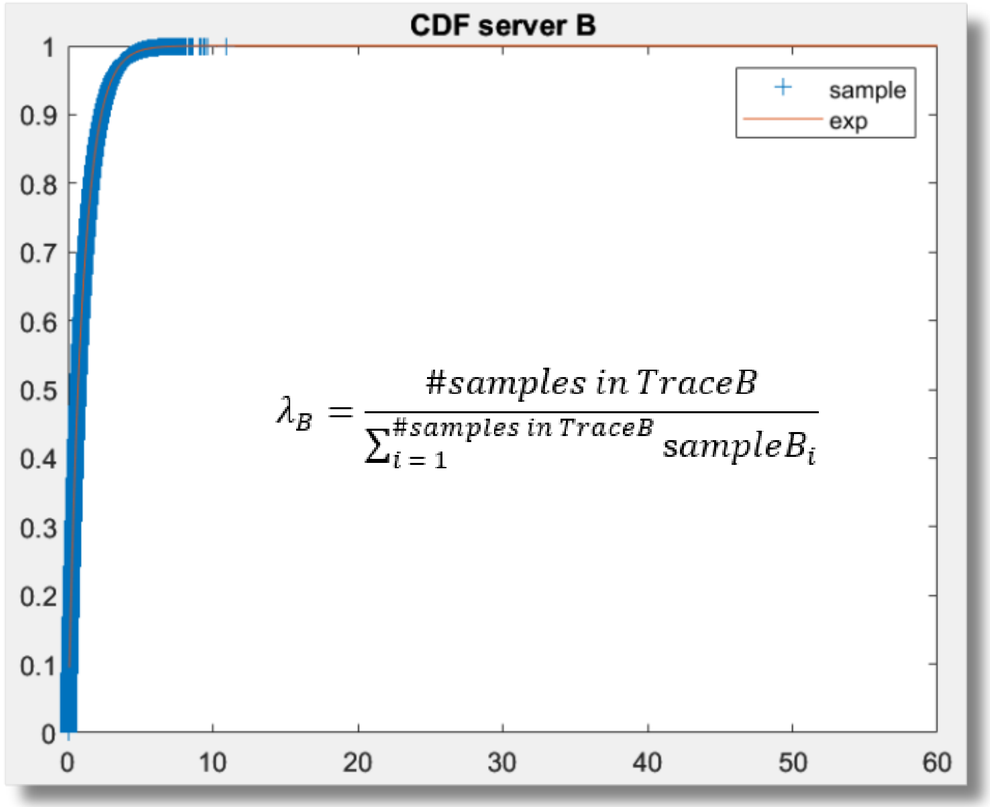
Data Fitting

Each Trace file has 50000 samples representing the execution time of the algorithm in milliseconds. To find the most suitable distribution to fit the data, the approximated **CDF function** was plotted for each Trace file. The plots suggested the possibility for the data to have been drawn by an **exponential distribution.** This insight was confirmed after deriving the **lambda parameters** from each Trace and plotting the corresponding CDFs against the approximated CDFs.

Data Fitting

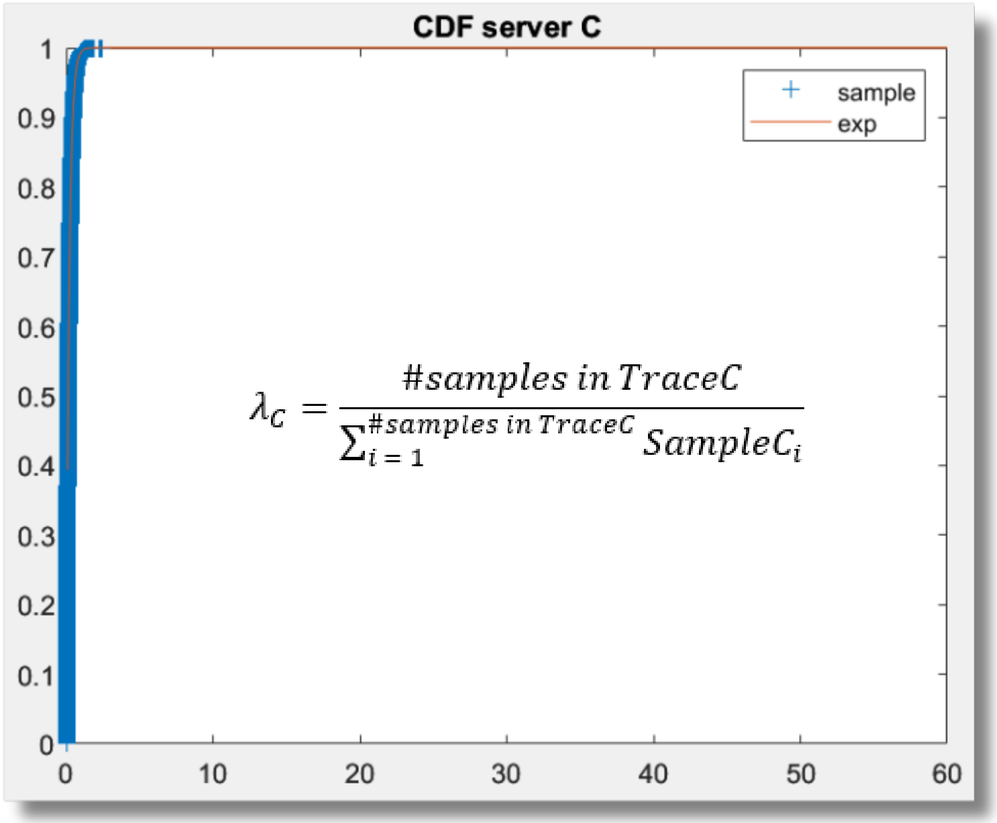


fitting stage A

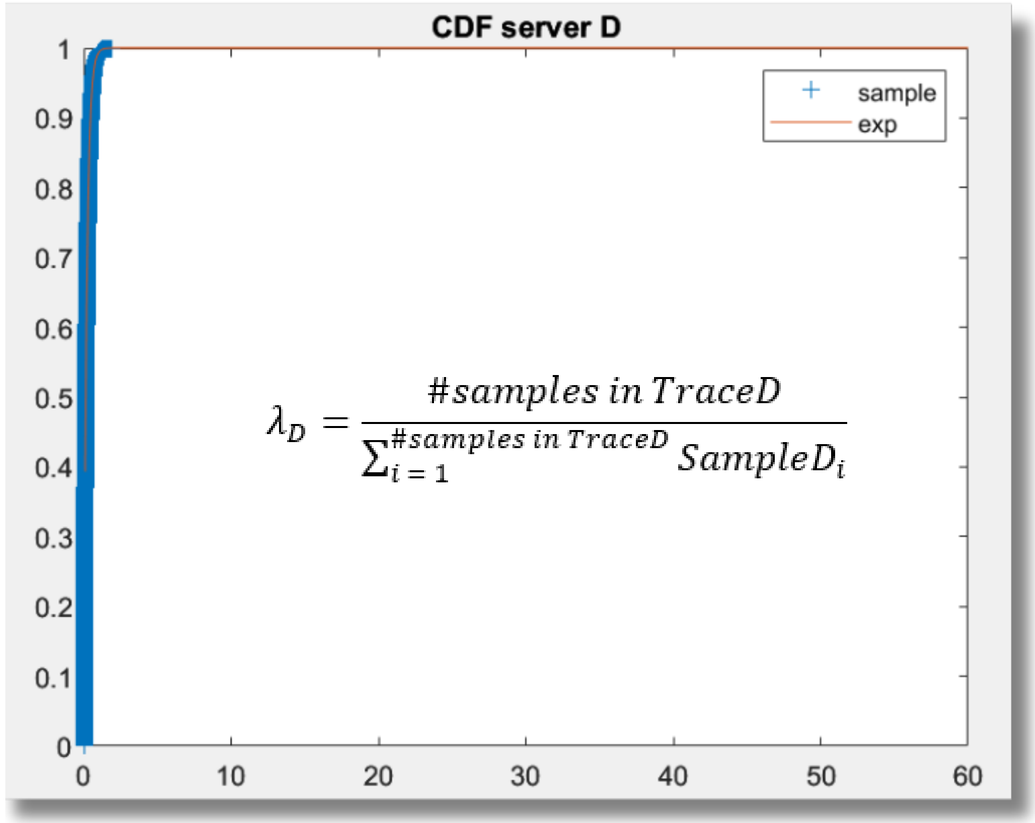


fitting stage B

Data Fitting

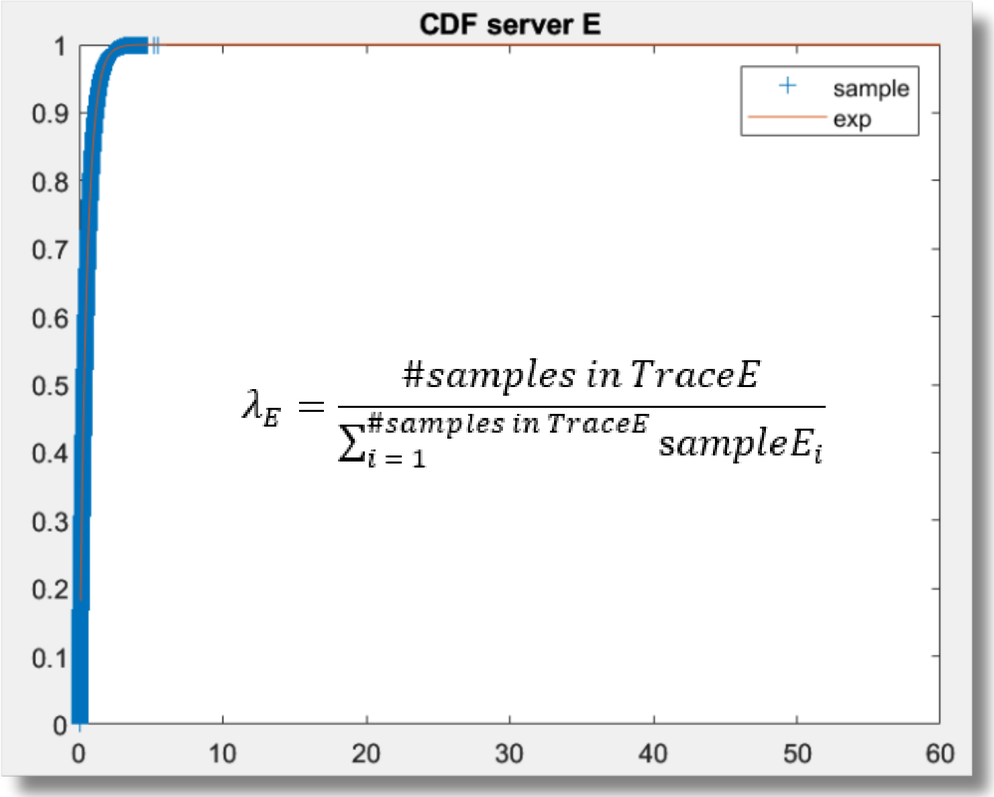


fitting stage C



fitting stage D

Data Fitting



fitting stage E

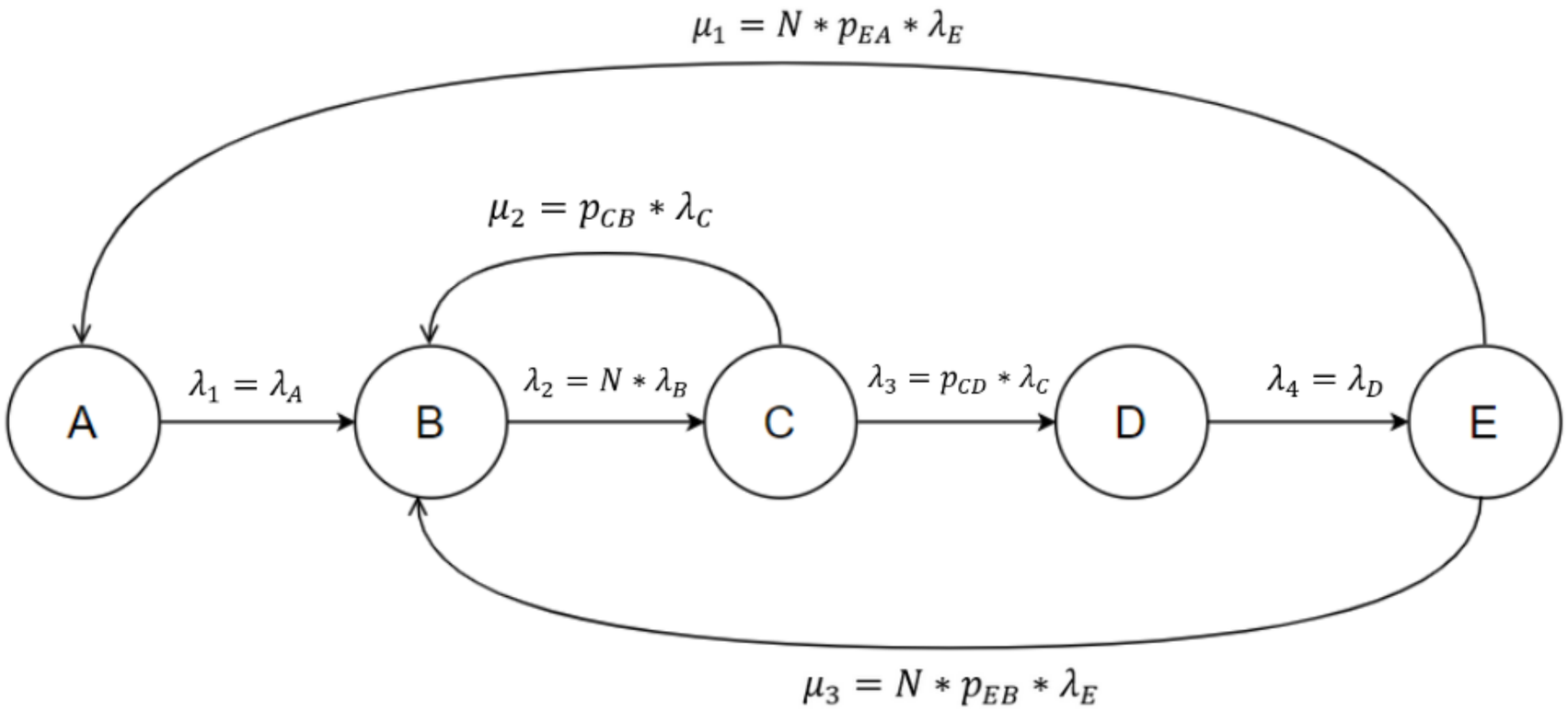
Model Decision

- Since each stage of the algorithm presents an **exponentially distributed service time**, the system can be modelled as a **Continuos Time Markov Chain (CTMC)**

- The system could also be modelled as a **Closed Queing Network** of **batch type** with one job

The desired performance indexes were calculated using **both models**, in order to have **more robust results.**

Corresponding State Machine



$N = number\ of\ cores$

$p_{CB} = 0.9$

$p_{CD} = 0.1$

$p_{EA} = 0.2$

$p_{EB} = 0.8$

Computation

Steady state distribution

$$u = 1 \quad 0 \quad 0 \quad 0 \quad 0$$

$$\pi = u * Q'^{(-1)}$$

Infinitesimal generator

$$Q = \begin{matrix} -\lambda_1 & \lambda_1 & 0 & 0 & 0 \\ 0 & -\lambda_2 & \lambda_2 & 0 & 0 \\ 0 & \mu_2 & -\mu_2 - \lambda_3 & \lambda_3 & 0 \\ 0 & 0 & 0 & -\lambda_4 & \lambda_4 \\ \mu_1 & \mu_3 & 0 & 0 & -\mu_1 - \mu_3 \end{matrix}$$

Q with normalization condition

$$Q' = \begin{matrix} 1 & \lambda_1 & 0 & 0 & 0 \\ 1 & -\lambda_2 & \lambda_2 & 0 & 0 \\ 1 & \mu_2 & -\mu_2 - \lambda_3 & \lambda_3 & 0 \\ 1 & 0 & 0 & -\lambda_4 & \lambda_4 \\ 1 & \mu_3 & 0 & 0 & -\mu_1 - \mu_3 \end{matrix}$$

Average throughput

$$E[X] = \sum_i \pi_i * \sum_{i \neq j} q_{ij} * x_{ij}$$

Transition matrix for throughput

$$X = \begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{matrix}$$

Solution

The steps in the previous slide were implemented in a Matlab script. The script was runned trying **different values** for the variable **N** (number of cores), starting from N = 1, in ascendent order, until an **average throughput** of more than 30 solutions/second was reached.

This value was reached with **N = 4**, which means that the **minimum number of cores** required to provide an average of more than 30 solutions per second is **4.**

$$E[X](4) = \sum_{i} \pi_i * \sum_{i \neq j} q_{ij} * x_{ij} = 34.3 \text{ solutions}/second$$

Solution

The last step is to find the **average utilization** of the CPU. The computation of this index follows directly from the number of cores of the CPU, which was found previously. Knowing that of all 5 stages of the algorithm, **only stage B and E** can be **fully parallelized**, while the **other stages** (A,C,D) would not obtain any benefit from parallelization, and would **run on a single core**, the formula for the total CPU utilization is the following:

$$U_{CPU} = \frac{cores_A}{cores_{tot}} + \frac{cores_B}{cores_{tot}} + \frac{cores_C}{cores_{tot}} + \frac{cores_D}{cores_{tot}} + \frac{cores_E}{cores_{tot}} =$$

$$= \frac{1}{4} + 1 + \frac{1}{4} + \frac{1}{4} + 1 = \frac{11}{4} = 2.75$$

While the average utilization can be computed as:

$$\overline{U}_{CPU} = \frac{U_{CPU}}{N} = \frac{2.75}{4} = 0.69$$

General Analysis

In this case the system is modelled as a **closed queuing network with one job**. In particular the system is a **separable queuing network**, therefore **analytical formulas** can be used to determine the performance indexes. To solve separable **Closed Queuing Networks** the formulas of the **Mean Value Analysis (MVA)** are required.

Another way in which the model can be solved is via **discrete event simulation**, but in this case the resulting indexes are provided as **confidence intervals.**

In both cases to solve the model the **Java Modelling Tool (JMT)** program was used. For the **MVA** the **JMVA** module of JMT was used, while for the discrete event simulation the **JSIMgraph** module of JMT was used.

MVA

To perform the **MVA**, JMVA needs the **number of visits** and the **average service times** for each station. Therefore both the number of visits and the service times need to be computed.

- Computation of the **visits**:

routing probability matrix

$$P = \begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0.9 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0.2 & 0.8 & 0 & 0 & 0 \end{matrix}$$

P with element corresponding to the reference station equal to 0

$$P' = \begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0.9 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0.8 & 0 & 0 & 0 \end{matrix}$$

Identity matrix

$$I = \begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{matrix}$$

vector with element corresponding to the reference station equal to 1

$$l = 1 \quad 0 \quad 0 \quad 0 \quad 0$$

$$v = l * (I - P)^{(-1)}$$

visits

MVA

- Computation of the average service times:

$$S_A = \frac{1}{\lambda_A} \qquad S_B = \frac{1}{N * \lambda_B} \qquad S_C = \frac{1}{\lambda_C} \qquad S_D = \frac{1}{\lambda_D} \qquad S_E = \frac{1}{N * \lambda_E}$$

Since stage **B** and **E** provide **full parallelization**, the **average service** time in these stages is **inversely proportional** to the **number of cores**. In order to save time in the MVA performed by JMVA, N will be set to 4, the value found by solving the CTMC.

MVA

After having found the number of visits and the average service time for each stage in the algorithm, we can provide these data as input to JMVA. On the right the results provided by JMVA show that the **system throughput** is **almost identical** to the one found **solving the CTMC.** For the **CPU utilization** the computation is **identical** to that of the **CTMC**.

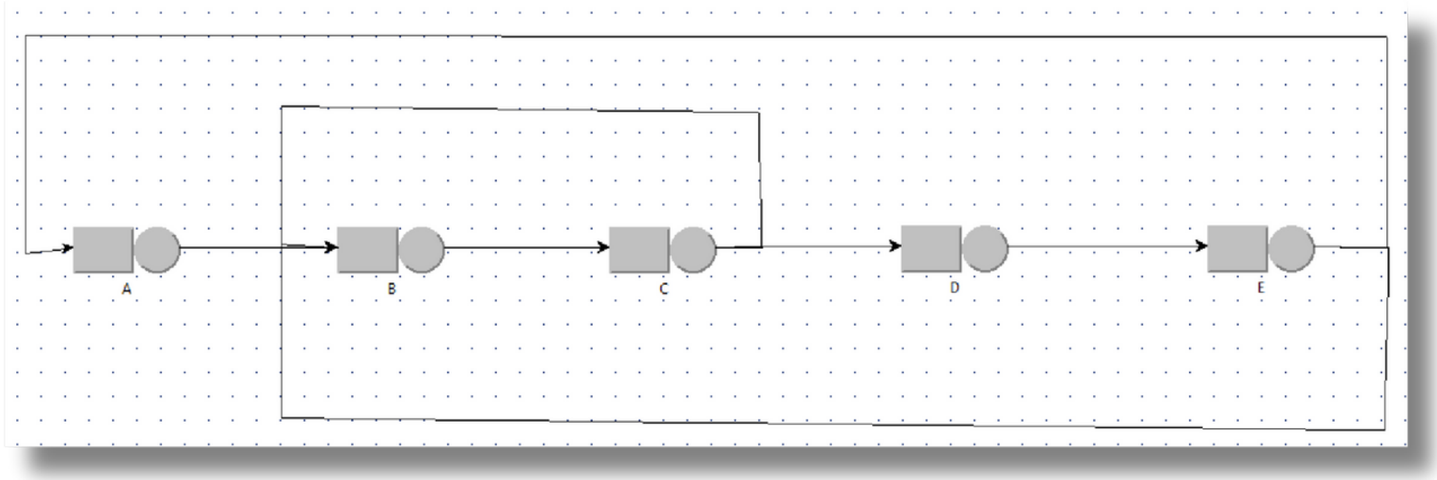
Throughput

Throughput of each class for each **Station**.

*	*Aggregate*	Class1
System	34.2971	34.2971
Station1	34.2971	34.2971
Station2	1714.85...	1714.8...

Results provided by JMVA

Discrete event simulation



Model drawn in JSIMgraph

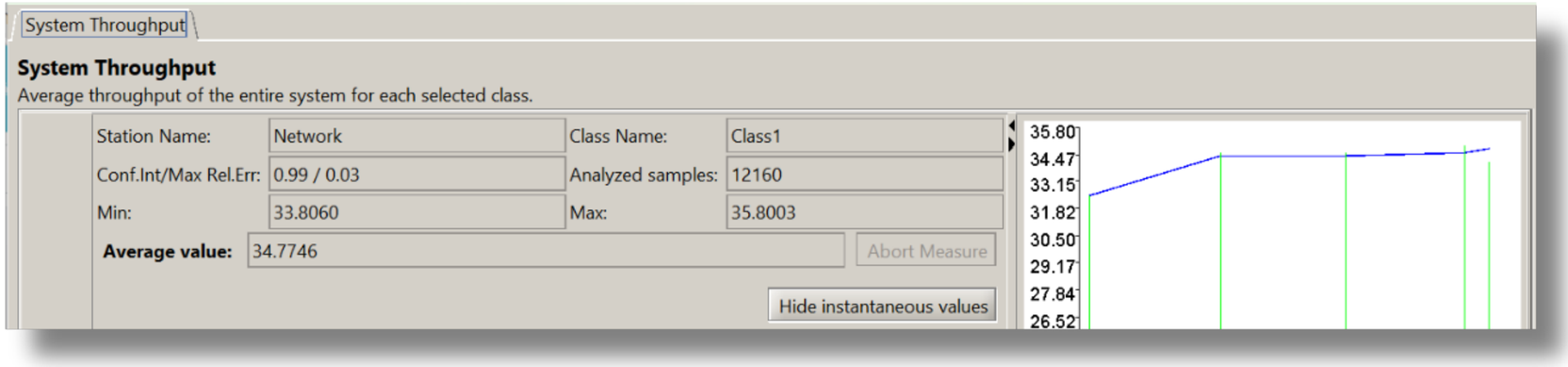
As mentioned previously, the **discrete event simulation** is performed by the JMT module **JSIMgraph**. Here the lambdas for the service time distributions in servers B and E were set respectively to:

$$N * \lambda_B \; and \; N * \lambda_E, \text{ with } N = 4$$

For the **CPU utilization** the computation is **identical** to that of the **CTMC** and **MVA**.

Discrete event simulation

Also in this case the **final solution** for the system throughput is **really close to the CTMC and MVA solutions**. In particular the confidence interval provided with **confidence level 0.99** and Maximum Relative error of 0.03 is: **33.8060** for the **minimum value** and **35.8003** for the **maximum value**. The **average value** is **34.7746**.



Results for the system throughput in JSIMgraph

Conclusions

Several methods (CTMC, MVA, discrete event simulation) were **used for the analysis** of the proposed **system** in order **to have** more **robust results**. Seeing at the final **results for each method**, it's possible to note that they are **consistent** among them. In conclusion to respond to the questions posed at the beginning of this project:

- The **minimum number of cores** to have a Throughput of 30 solutions per second is: **4**
- The average **CPU utilization** is: **0.69**