Teaching scheduling and routing algorithms through animations with the JMCH component of JMT

Marco Gribaudo[1,*,†], Giuseppe Serazzi[1] and Lorenzo Torri[1]

[1]Dip. di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, via Ponzio 34/5, 20133 Milano, Italy

Abstract

This paper focuses on describing a new feature of the *Java Modelling Classroom Helper* (JMCH), one of the tools of the Java Modelling Tools (JMT) suite, which aims to provide better educational potential in teaching scheduling and routing policies within queueing networks. A controlled animation scenario has been implemented that allows users to follow the behavior of the algorithms via graphical animation through an intuitive user interface and evaluate their performance. Users are gradually introduced to the complexity of the scheduling algorithms, and learn the impact of each policy decision as they go through the implemented models. Several scheduling and routing algorithms are considered together with four distributions of service times and arrival processes. The results of the animated model are summarized in tables and compared with the values of the correspondent metrics obtained by simulation with JSIM.

Keywords

routing algorithms, job scheduling, teaching aid, queueing networks

1. The problem approached

The scheduling algorithms play a fundamental role in the performance of digital systems. These algorithms are used at various levels of a system, from a single resource to a complete infrastructure. Although scheduling policies can be rigorously described using formal languages, teaching them is not an easy task.

The main source of complexity is due to the fact that their behavior strongly depends not only on the implemented policies, but also on the statistical properties of service times and arrival process. As a consequence of this dependence, it is impossible to evaluate the effectiveness of different policies without knowing the statistical characteristics of these variables. Thus, the best scheduling algorithm for a given problem cannot be identified once and for all, but must be found considering the workload to be processed. The results may be different as the workload changes.

Furthermore, fluctuations in service requests and interarrival times are directly transferred to the results provided by the different algorithms.

Interactions between policy logics, characteristics of service requests, and arrival processes increase the difficulty of teaching the behavior of scheduling algorithms.

To reduce the complexity of this task we improved the JMCH, *Java Modelling Classroom Helper*, one of the JMT tools [1, 2, 3], implemented originally to support the teaching of the Markov Chains corresponding to a single queueing station. Two new features, focused on the scheduling and the routing algorithms, have been added [4]. Basically, a *controlled animation scenario* has been designed to allow users to interact with the algorithms during their behavior. Through a user-friendly GUI it is possible to analyze the effects of the decisions taken by the algorithms step by step. Users may investigate on the impact of different scheduling strategies on the performance and, using different windows, may compare the behavior of various algorithms. Five non-preemptive scheduling algorithms of a single queue station, three routing algorithms of a network with three queue stations, four distributions of service times, and four different arrival processes have been implemented. There are several parameters

QualITA 2025: The Fourth Conference on System and Service Quality, June 25 and 27, 2025, Catania, Italy

†These authors contributed equally.

marco.gribaudo@polimi.it (M. Gribaudo); giuseppe.serazzi@polimi.it (G. Serazzi); lorenzo.torri@virgilio.it (L. Torri)

00000-0002-1415-5287 (M. Gribaudo)

available to control both the animation and the executions of the jobs, such as the number of requests generated by the source, the number of dropped requests, the animation length, the animation speed, the seed of the random number generator.
Furthermore, the animated models are also solved with the JSIM simulator and the results are summarized in rows of a table, allowing an immediate comparisons with those obtained from animations.
The animation characteristics of the M/M/c/K queuing models are described in Sect.3.3, scheduling algorithms are described in Sect.3.1 while those of the routing algorithms are described in Sect.3.2.

2. JMT

The Java Modelling Tools (JMT) is a open source suite of six tools that aims to provide a comprehensive framework for performance engineering and capacity planning of digital infrastructures. Analytical and simulation techniques are applied to solve the Queueing Networks and Petri Nets models implemented with the tools [2].

The current stable version of the suite includes six Java applications:

1. JSIMgraph - Queueing networks and Petri nets simulator with graphical user interface
2. JSIMwiz - Queueing network and Petri net simulator with wizard-based user interface
3. JMVA - Mean Value Analysis and Approximate solution algorithms for Queueing network models
4. JABA - Asymptotic Analysis and bottlenecks identification of Queueing network models
5. JWAT - Workload characterization from log data
6. JMCH - Modelling Classroom Helper for learning scheduling algorithms through animations

Fig.1 shows the start-up screen of the JMT tools, where the users can select the specific components.

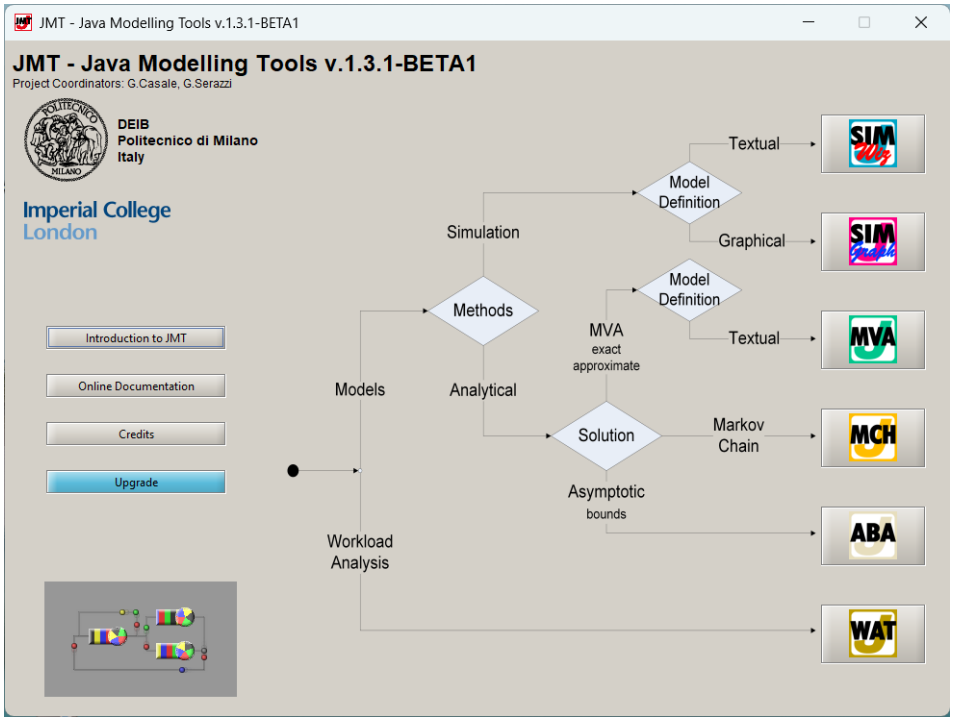


Figure 1: The JMT tool selection menu.

The most popular tool of JMT is JSIMgraph, since it allows to define and analyze models in a user friendly way. It supports several modelling primitives, allowing to study multiformalism models [5, 6] with the direct interaction of Petri Nets, Queueing Network, as well as custom primitives to model fork-join and distributed applications. Multiformalism models are very attractive [7] since they allow

users to combine the power of different modelling languages, exploiting the best features that each one can offer [8]. Figure 2 shows an example of a multiformalism JSIMgraph model for dynamically routing arriving request spikes [3] to a dedicated Spike Server.

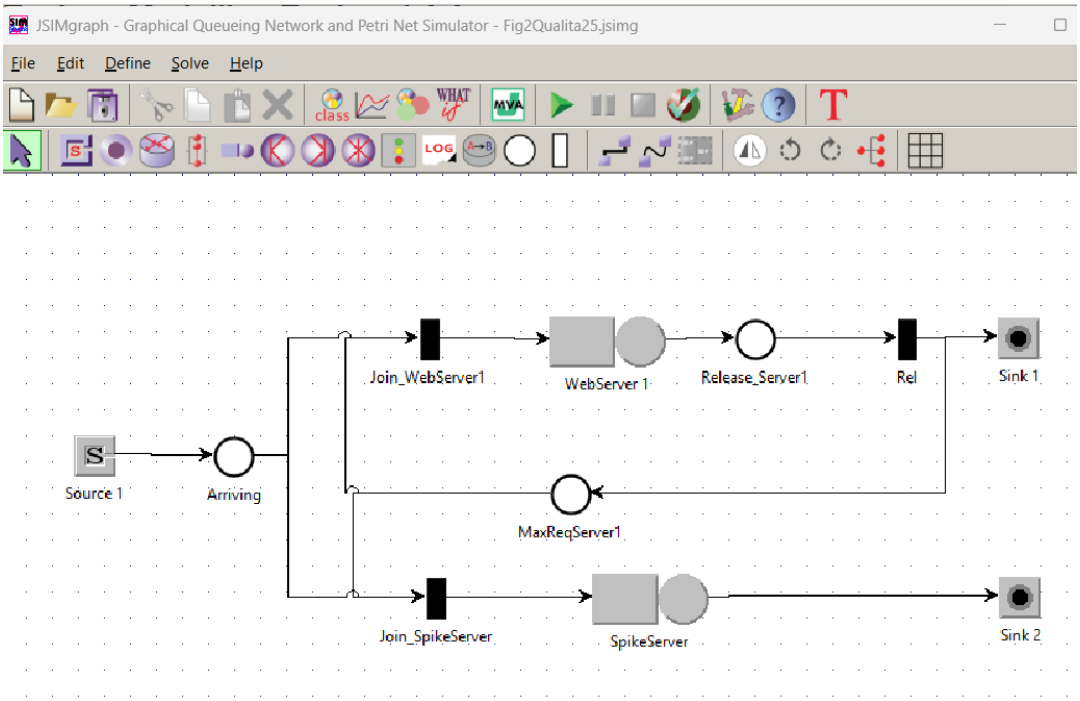


Figure 2: A multiformalism JSIMgraph model for the dynamic routing of traffic spikes.

3. JMCH - The global structure

The starting screen of JMCH (Fig.3) allows the selection of the algorithms that can be analyzed via animation: five *scheduling algorithms* of a single queueing station, three *routing algorithms* of a network of three queueing stations, and the *Markov Chain states* of a queueing station with different configurations.

3.1. Scheduling Algorithms

The system considered in this section consists of a single queueing station with one queue of requests (also referred to as *req* or *job*) waiting for the service and one or more servers. When a server is free one request is selected from the queue and its execution starts. The decision of which request must be selected from the queue depends on the scheduling discipline. We implemented four non-preemptive algorithms (`FCFS First Come First Served`, `LCFS Last Come First Served`, `SJF Shortest Job First`, `LJF Longest Job First`) and the `PS Processor Sharing`. The graphical representation of the movement of requests in the station required the introduction of some limitations in the system architecture. First, we limited the capacity of the queue to *five* requests waiting for the execution. Thus the maximum number of requests in the station is five plus the requests in the servers. For simplicity, in what follows we consider a station with one server. Requests arriving when the queue is full are *dropped*. The use of the drop policy allows to expand the types and characteristics of the arrival processes that can be considered. In this way, even arrivals that saturate the system can be shown by the animation.
Four interarrival times distributions are available: `Exponential`, `Uniform`, `Deterministic`, `Hyper-exponential`. The same distributions are available for *service times*. A screenshot of the animation of a queueing station is shown in Fig.4. According to the parameters, the selected policy is FCFS, the

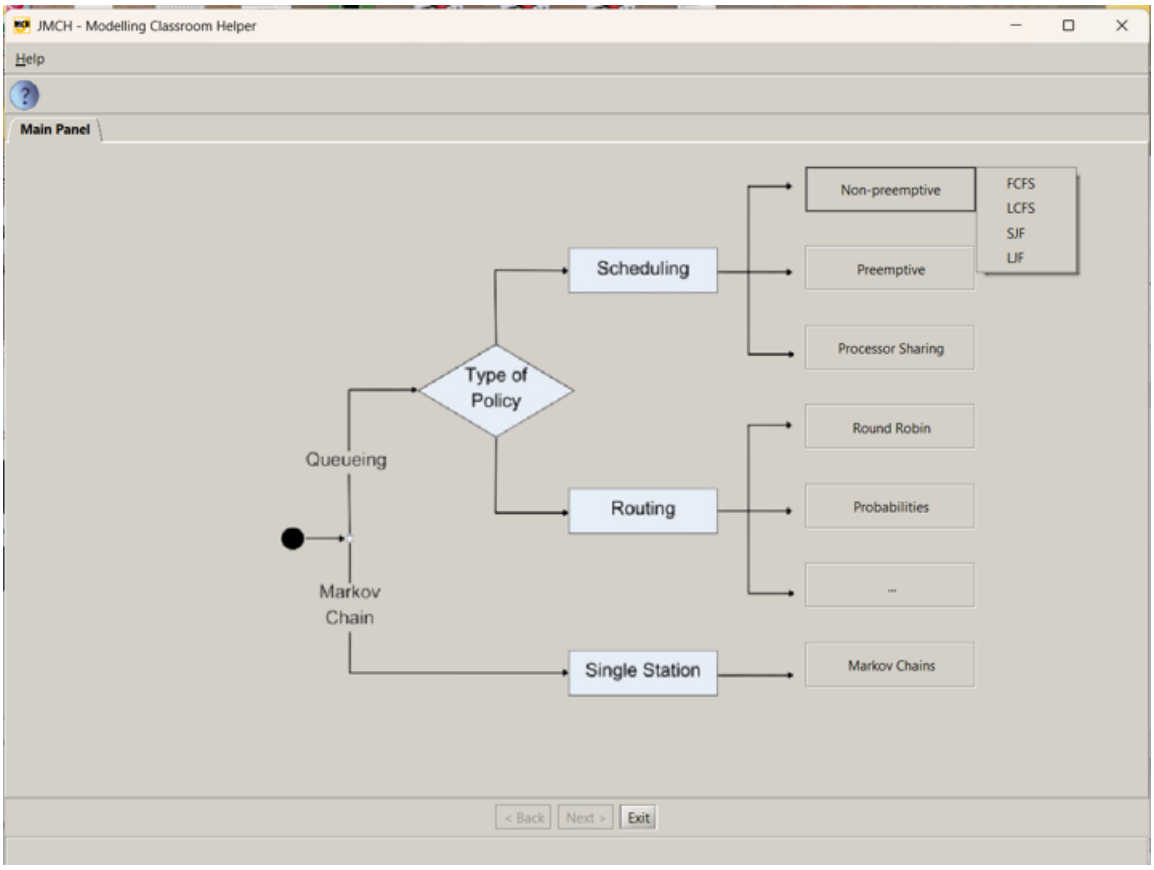


Figure 3: The selection window of the JMCH

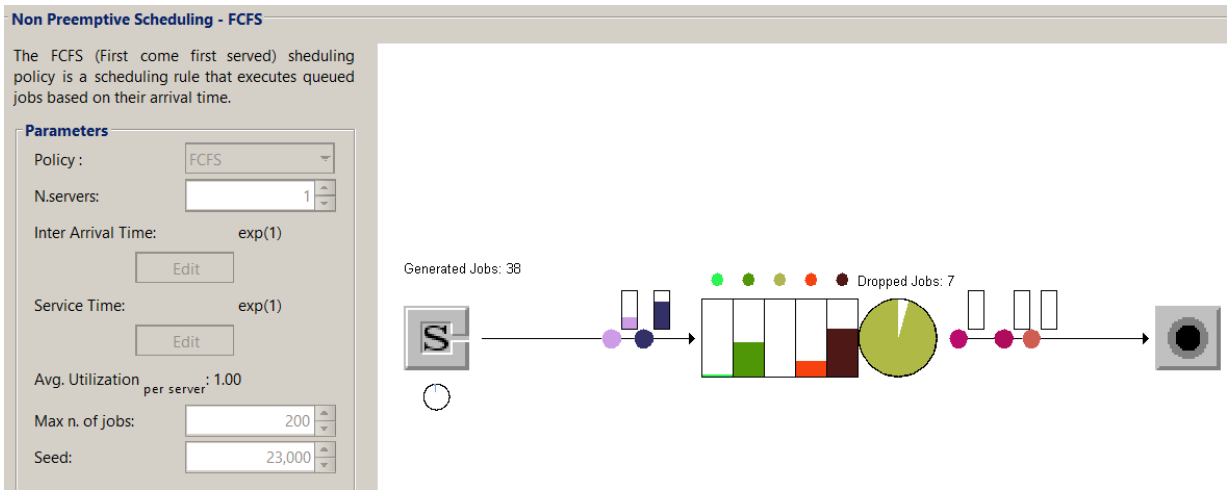
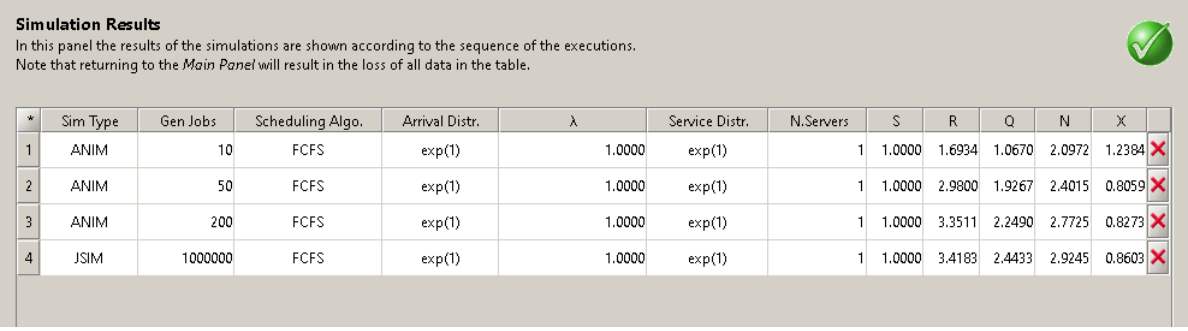


Figure 4: A screenshot of the animation of a queueing station with the FCFS scheduling algoritms.

interarrival times and service times are exponentially distributed with mean values 1, the max number of requests that will be animated is 200, and the random number generator seed is 23000. The screenshot was taken after 38 req were generated, 7 of which were dropped. Six buttons are available in a Button Toolbar to control the animation: `Play`, `Pause`, `Restart`, `Single Step`, `Increase/Decrease animation speed`. `Play` and `Pause` let the execution of the animation run at a speed proportional to the simulation times, while `Single Step` allows to focus on the way in which the system will evolve in the current configuration: in particular, it generates both an anticipation of what the student thinks will happen next and the confirmation or correction of the hypothesized behavior. A request is represented

by a circle filled with color and a rectangle with an area filled with the same color whose size represents its service demand. As execution progresses, the circle and rectangle move along the model connections from source to sink. The colored area of the server represents the fraction of the server capacity utilized. The results obtained from the animation of a JMCH run and those of a correspondent model simulated with JSIM are reported in a `Results table`. Fig.5 shows some metrics obtained with three runs of the same model (ANIM) with different animation lengths (10, 50, 200 req) and those of the corresponding JSIM model. Clearly, as the animation length increases the values of the ANIM metrics converge to those of the JSIM.



*	Sim Type	Gen Jobs	Scheduling Algo.	Arrival Distr.	λ	Service Distr.	N.Servers	S	R	Q	N	X	
1	ANIM	10	FCFS	exp(1)	1.0000	exp(1)	1	1.0000	1.6934	1.0670	2.0972	1.2384	✕
2	ANIM	50	FCFS	exp(1)	1.0000	exp(1)	1	1.0000	2.9800	1.9267	2.4015	0.8059	✕
3	ANIM	200	FCFS	exp(1)	1.0000	exp(1)	1	1.0000	3.3511	2.2490	2.7725	0.8273	✕
4	JSIM	1000000	FCFS	exp(1)	1.0000	exp(1)	1	1.0000	3.4183	2.4433	2.9245	0.8603	✕

Figure 5: Results of three animations (ANIM) of the same model with different number of generated jobs and those obtained from the corresponding JSIMgraph model.

3.2. Routing Algorithms

The focus of this section is on *routing algorithms*. Due to the limitations introduced by the graphical representation of routing algorithms, we only consider systems consisting of three queueing stations. We assume that the routing decision is made immediately after a job enters the system, and that the time required to reach the selected station is negligible. The stations have the same characteristics and are similar to the one described in the Scheduling Algorithms section. The scheduling discipline adopted in all of them is non-preemptive FCFS (`First Come First Served`). Moreover, each station has a single server. In the graphical representation of the model, a new component is introduced: the `Router`. Its role is to forward incoming jobs from the `Source` to one of the three outgoing arcs, each connected to a queueing station, according to a predefined policy. Three routing algorithms are currently implemented (`RR Round Robin`, `JSQ Join Shortest Queue`, `Probabilities`). In the latter case, the user can associate a different routing probability to the paths connecting the three stations. As in the case of the scheduling algorithms, four distributions of interarrival times and service times are available: `Exponential`, `Uniform`, `Deterministic`, `Hyperexponential`. A screenshot of the animation of a three-station model is shown in Fig.6.
The selected routing algorithm is `Probabilities`. According to the parameters, the three probabilities associated with each outgoing arc from the router, starting from the top and moving clockwise, are 0.2, 0.5, and 0.3, respectively. The interarrival times follows an Exponential distribution with mean value 1, and the service time also follows an Exponential distribution with mean 5. The system operates under saturation conditions. The maximum number of animated requests is set to 2000, and the random number generator seed is 23000. The screenshot was taken after 64 generated jobs, 26 of which were dropped.

The same six buttons of the `Button Toolbar` described for the scheduling policy, are available also in this case to control the animation: `Play`, `Pause`, `Restart`, `Single Step`, `Increase/Decrease` animation speed.

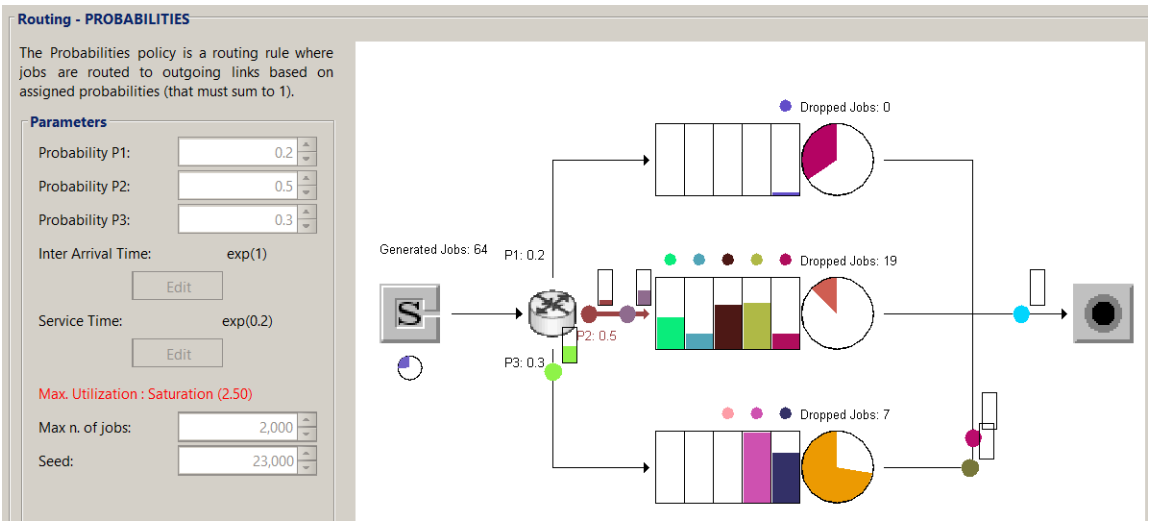


Figure 6: Screenshot of an animation of a `Probabilistic` routing algorithm between three queue stations.

3.3. The Markov chains analyzer

The JMCH provides a graphical representation of the Markov Chain states corresponding to a single queue station model, including single-server stations with finite (M/M/1/k) or infinite (M/M/1) queue size, and multi-server stations with unlimited (M/M/c) or limited (M/M/c/k) queue size. It is possible to change the arrival rate λ and the service time S of the station at runtime. Users can get visual perception of traffic bursts and their effects on queue length and server utilization. The exact analytical results are computed and also displayed for comparison purposes.

The application starts by asking the user what type of queue station needs to be analyzed. The selection screen is shown in Fig.7.

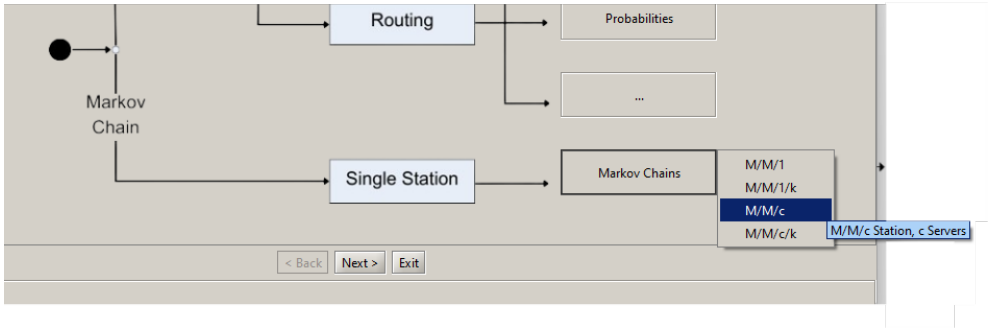


Figure 7: The selection process of the specific M/M/c/K queueing station.

After the selection, the corresponding states of the birth-death process are represented, see Fig.8 , and the animation of the sequence of states visited during the simulation is shown. The following parameters can be changed dynamically via sliding cursors: *arrival rate* λ, *service time* S, *station capacity* k. The arrival rate and service times are exponentially distributed. Among the performance indices computed and/or displayed are: *mean number of customers* in the station, *throughput*, *utilization*, *probability of each state* of the Markov Chain. The current state of the Markov Chain is colored in red and the state probabilities are shown in green.

A log file is generated at each simulation with the following data: *Customer ID* (the id assigned to each customer), *Arrival Time* (the instant of time of the creation of the customer), *Start Execution* (the instant of time in which the customer start to be executed from one of the servers), *Server ID* (it shows the id of the server if there are more than one), *Exit system* (the instant of time in which the customer exits from the system).

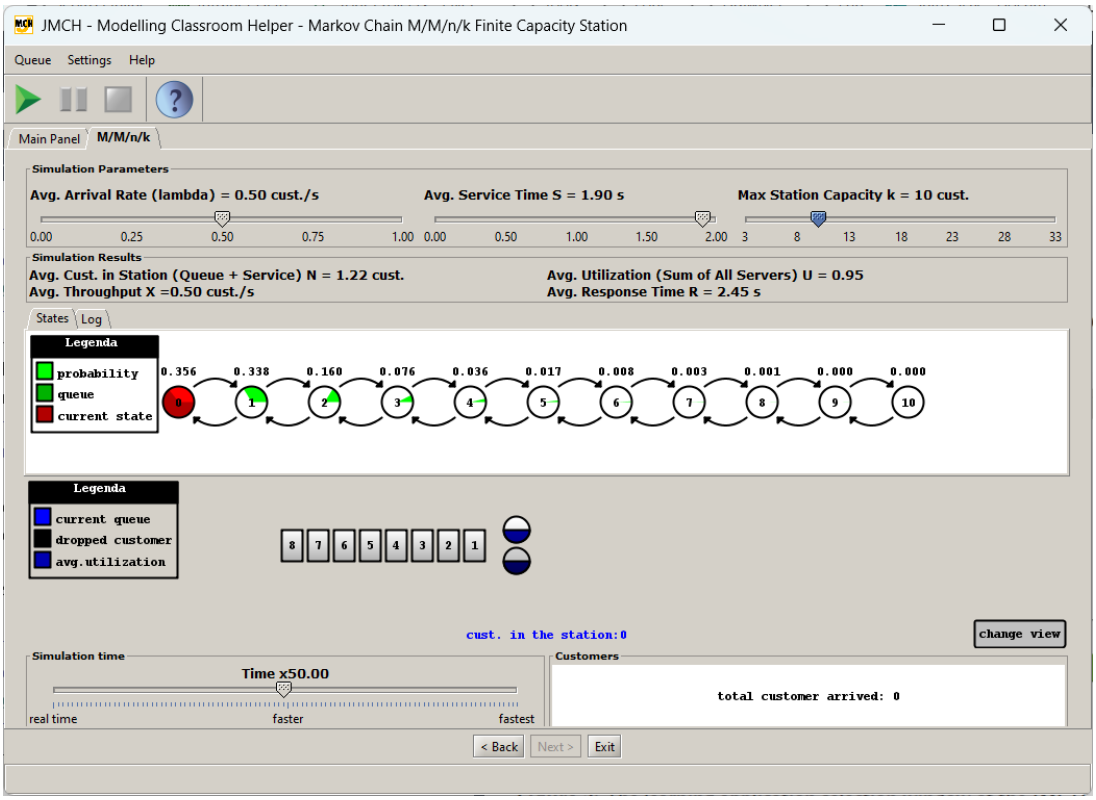


Figure 8: The animation of the selected M/M/2/10 queueing station.

4. Effectiveness of the animation technique

The proposed tool can be used to create some lessons for the performance evaluation course, where the tool is provided to students along with some printed materials that can be useful during the session. Fig.9 shows a typical workflow of a lesson that use JMCH to teach scheduling policies, routing algorithms, and M/M/c/K queue stations. First, the exercise session is briefly described, and the printed material is distributed to the students. This can be either on paper or on a web resource, depending on the organization of the classroom. Subsequently the students download and install the tool (unless this was already used in a previous lecture), opens them and try to familiarize with both the tool and the provided material.

Fig.10 shows some of the slides given to the students: part a) shows an example of the instructions that can be given to download and install the tool. A general introduction to the tool interface could be given, as shown in Fig.10b). Then the session continues with scientific content part, where instruction to set-up the parameters and their explanations are given, as shone in Fig.10c). Also, some content explaining how to run and control the execution of the animation can be useful, as shown in Fig.10d).

The experimentation by the students will continue for about 20 minutes to 1 hour, considering that each individual topic (i.e. scheduling policies, routing algorithms, and M/M/c/K models) should require around 20 minutes. During this time, the teacher will wander around the class, giving brief prompts to individual students. He helps them focus on the most important concepts that need to be considered, also showing which parameters can be changed to achieve interesting or unexpected effects.

The lesson concludes by assigning students some simple exercises to immediately verify what they have learned. Fig.11 shows an example of some questions. In this case, due to the special nature of the exercise session, it might be worth not to use the evaluation of these exercises as part of the final mark, to reduce the pressure and make the experience more enjoyable.

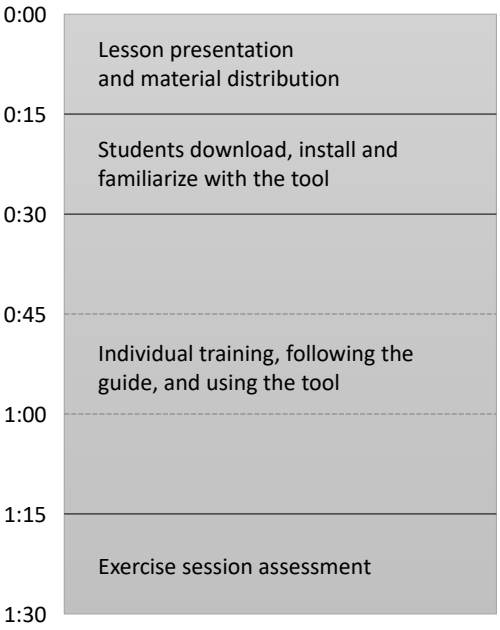


Figure 9: A typical workflow of a lesson which exploits JMCH to teach the corresponding subjects.

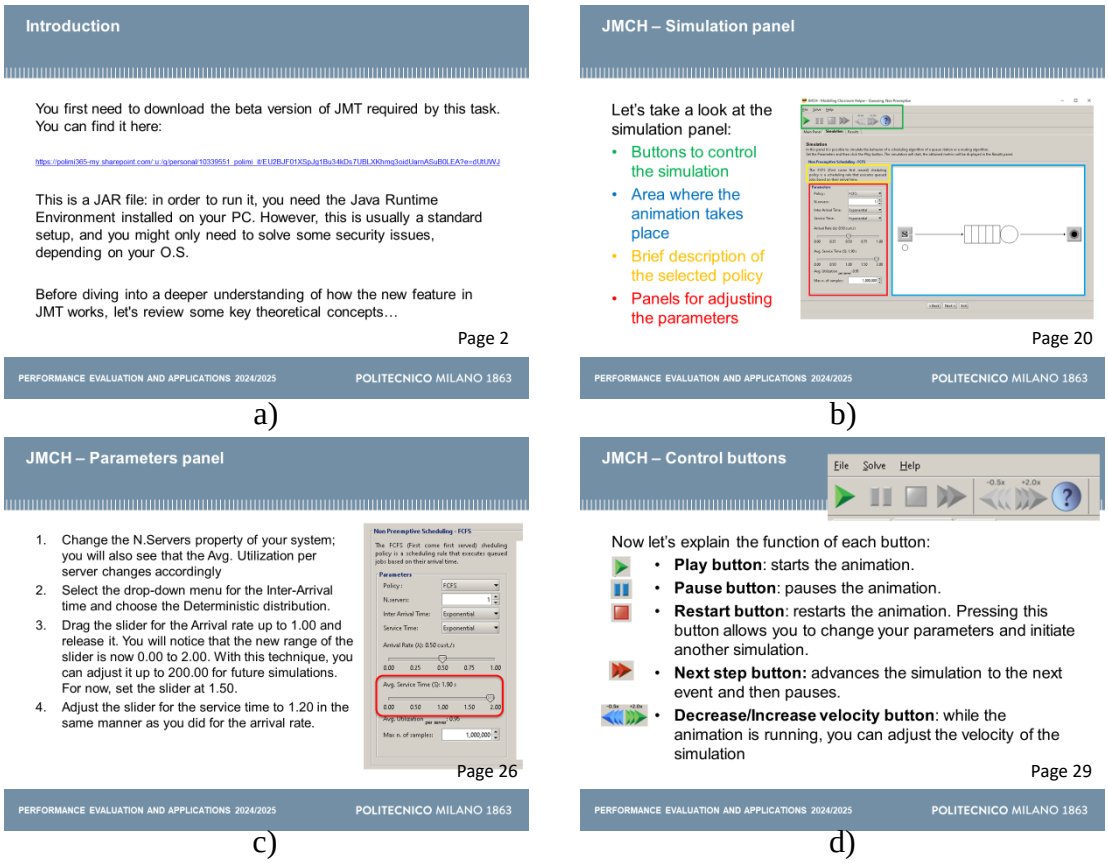


Figure 10: Example of material provided to students.

5. Conclusion and Future developments

The final goal of this work is to improve the JMCH so that it can be used as a *testbed* for designing and evaluating scheduling and routing algorithms. Possible extensions that are currently being implemented

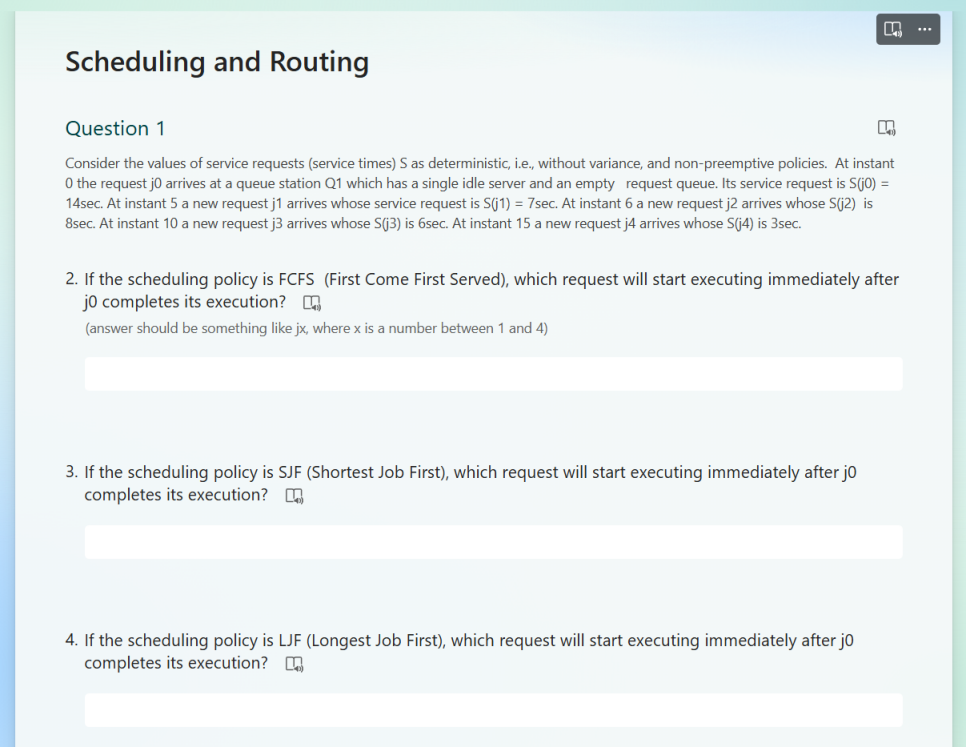


Figure 11: Example of exercises used to assess the learning level of students in a JMCH-based session.

or planned are: a more complete set of distributions for both the interarrival and the service times (see the ones supported by JSIM), a larger set of different scheduling and routing algorithms, as well as a decrease in the number of constraints that limit current system architectures that can be considered. Another important feature that will be implemented is the possibility to accept as input a log file containing a sequence of interarrival and service times submitted by users (like the `Replayer` feature of JSIM). This will enable the use of collected data files in real systems.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] M. Bertoli, G. Casale, G. Serazzi, Jmt: Performance engineering tools for system modeling, SIGMETRICS Perform. Eval. Rev. 36 (2009) 10 – 15. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-77950493522&partnerID=40&md5=0338ca2ad38c9b65548ca6d980e8ed78.

[2] Java Modelling Tools JMT: performance engineering tools for system modelling, https://jmt.sourceforge.net/, 2025. [download open source].

[3] G. Serazzi, Performance engineering: Learning through applications using JMT, 2023. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85197189480&doi=10.1007%2f978-3-031-36763-2&partnerID=40&md5=65edd486fdab00142f889a357c09d2d1. doi:10.1007/978-3-031-36763-2.

[4] L. Torri, JMCH: Application of Animation Techniques for Teaching Scheduling and Routing Algorithms with JMT, 2024.

[5] E. Barbierato, M. Gribaudo, G. Serazzi, Multi-formalism models for performance engineering 12 (2020). URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85082960422&doi=10.3390%

2ffi12030050&partnerID=40&md5=b86dcb80c7974534ded37b8e74bb08bf. doi:10.3390/fi12030050, all Open Access, Gold Open Access, Green Open Access.

[6] M. Iacono, M. Gribaudo, E. Barbierato, Exploiting multiformalism models for testing and performance evaluation in simthesys, 2011, p. 121 – 130. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84897446724&doi=10.4108%2ficst.valuetools.2011.245727&partnerID=40&md5=4fab40f30e2a1d9188f93219c922ef11. doi:10.4108/icst.valuetools.2011.245727, all Open Access, Bronze Open Access.

[7] M. Iacono, M. Gribaudo, Element based semantics in multi formalism performance models, 2010, p. 413 – 416. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-78049499120&doi=10.1109%2fMASCOTS.2010.54&partnerID=40&md5=1724d38465a91c64bdbf574e87789a8b. doi:10.1109/MASCOTS.2010.54.

[8] M. Iacono, E. Barbierato, M. Gribaudo, The simthesys multiformalism modeling framework 64 (2012) 3828 – 3839. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84870243255&doi=10.1016%2fj.camwa.2012.03.009&partnerID=40&md5=f0864721bc1643b9307366b5705e77ad, all Open Access, Bronze Open Access.