POLITECNICO DI MILANO

Performance Evaluation and Applications

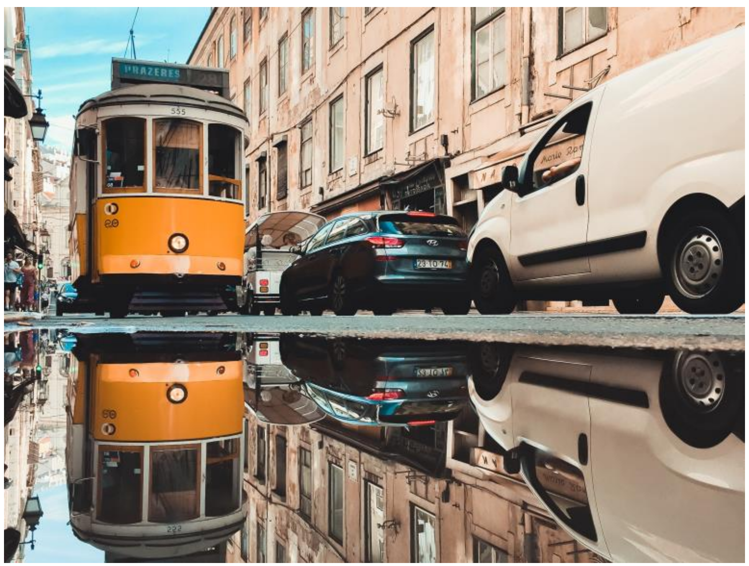


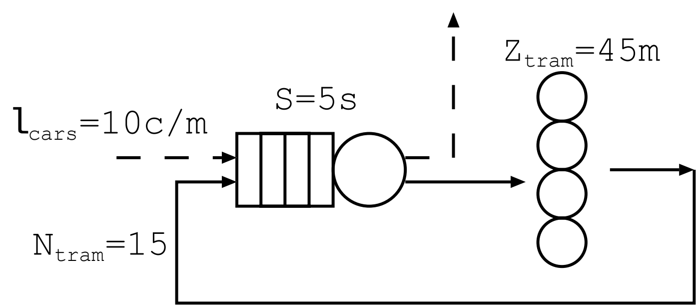


Solution of Multi-class models

Motivating example

A street is shared by both trams and regular cars. Trams returns to the same street section after an exponentially distributed amount of time, with an average of 45 minutes. A total of N_T = *15* trams run on that line. In the peek hour, cars arrives at a rate λ_C = *10* cars / minute. The crossing time of the street (when there is no queue) is 5 sec. Which is average time between the passage of two trams?





Closed and Mixed Multi-class models

Analytical solutions exists also for open, closed and mixed multi-class models.

Except for open models, their implementation is however quite complex, and we will resort to tools such as the JMVA module of JMT to use ready-made solutions of the considered systems.

We will briefly describe the relevant techniques to showcase how such implementations work.

Open models

The relation between residence time and number of jobs found at the arrival is valid for each class *c* in multi-class models.

$$R_{kc}(\lambda_1, \ldots) = \begin{cases} D_{kc} \cdot (1 + A_{kc}(\lambda_1, \ldots)) & \text{(queue)} \\ D_{kc} & \text{(delay)} \end{cases}$$

Again, for open models, the number of jobs at the station found at the arrival is equal to the average number of jobs in the long run.

This number however counts all the jobs of all classes in the station, and it is independent from the class.

$$A_{kc}(\lambda_1, \ldots) = N_k(\lambda_1, \ldots) = \sum_c N_{kc}(\lambda_1, \ldots) = \sum_c \lambda_c \cdot R_{kc}(\lambda_1, \ldots)$$

Open models

For open models, we can then derive R_{kc} from the expression:

$$R_{kc}(\lambda_1, \ldots) = D_{kc} \cdot \left(1 + \sum_{c'} \lambda_{c'} \cdot R_{kc'}(\lambda_1, \ldots)\right)$$

This time however R_{kc} cannot be computed immediately since it is embedded in a summation in the right hand side, which considers the residence time at the same station for the other classes c'.

Open models

However, at each station *k* the ratio between any two residence times $R_{kc'}$ and R_{kc}, of two classes *c'* and *c* have a fixed proportion:

$$\frac{R_{kc'}(\lambda_1, \dots)}{R_{kc}(\lambda_1, \dots)} = \frac{D_{kc'} \cdot \cancel{\left(1 + \sum_{c''} \lambda_{c''} \cdot R_{kc''}(\lambda_1, \dots)\right)}}{D_{kc} \cdot \cancel{\left(1 + \sum_{c''} \lambda_{c''} \cdot R_{kc''}(\lambda_1, \dots)\right)}}$$

In particular they are proportional their respective demands:

$$R_{kc'}(\lambda_1, \dots) = \frac{D_{kc'}}{D_{kc}} \cdot R_{kc}(\lambda_1, \dots)$$

Open models

We can exploit this to compute the average response time for a class *c* job at a station *k*.

$$R_{kc}(\lambda_1,\dots) = D_{kc} \cdot \left(1 + \sum_{c'} \lambda_{c'} \cdot R_{kc'}(\lambda_1,\dots)\right)$$

$$R_{kc'}(\lambda_1,\dots) = \frac{D_{kc'}}{D_{kc}} \cdot R_{kc}(\lambda_1,\dots)$$

$$R_{kc}(\lambda_1,\dots) = D_{kc} \cdot \left(1 + \sum_{c'} \lambda_{c'} \cdot \frac{D_{kc'}}{D_{kc}} \cdot R_{kc}(\lambda_1,\dots)\right)$$

$$R_{kc}(\lambda_1,\dots) \cdot \left(1 - \sum_{c'} \lambda_{c'} \cdot D_{kc'}\right) = D_{kc}$$

$$R_{kc}(\lambda_1,\dots) = \frac{D_{kc}}{1 - \sum_{c'} U_{kc'}(\lambda_1,\dots)} = \frac{D_{kc}}{1 - U_k(\lambda_1,\dots)}$$

Open models

To summarize:

$$\textit{processing capacity}: \max_k \left\{ \sum_{c=1}^{C} \lambda_c D_{c,k} \right\} < 1$$

$$\textit{throughput}: X_c(\vec{\lambda}) = \lambda_c$$

$$\textit{utilization}: U_{c,k}(\vec{\lambda}) = \lambda_c D_{c,k}$$

$$\textit{residence time}: R_{c,k}(\vec{\lambda}) = \begin{cases} D_{c,k} & \text{(delay)} \\ \dfrac{D_{c,k}}{1 - \sum_{j=1}^{C} U_{j,k}(\vec{\lambda})} & \text{(queueing)} \end{cases}$$

$$\textit{queue length}: Q_{c,k}(\vec{\lambda}) = \lambda_c R_{c,k}(\vec{\lambda})$$

$$= \begin{cases} U_{c,k}(\vec{\lambda}) & \text{(delay)} \\ \dfrac{U_{c,k}(\vec{\lambda})}{1 - \sum_{j=1}^{C} U_{j,k}(\vec{\lambda})} & \text{(queueing)} \end{cases}$$

$$\textit{system response time}: R_c(\vec{\lambda}) = \sum_{k=1}^{K} R_{c,k}(\vec{\lambda})$$

$$\textit{average number in system}: Q_c(\vec{\lambda}) = \lambda_c R_c(\vec{\lambda}) = \sum_{k=1}^{K} Q_{c,k}(\vec{\lambda})$$

Closed models

For closed models, $A_{kc}(\ldots)$ corresponds to the average number of jobs at the considered station k when in the system there is one less job of the target class c.

$$A_{kc}(N_1, \ldots) = N_k(N_1, \ldots, N_c - 1, \ldots)$$
$$R_{kc}(N_1, \ldots) = D_{kc} \cdot (1 + N_k(N_1, \ldots, N_c - 1, \ldots))$$

From the residence time of each class at each station, we can determine the system response times and the throughputs per class.

$$R_c(N_1, \ldots) = \sum_{k=1}^{K} R_{kc}(N_1, \ldots)$$
$$X_c(N_1, \ldots) = \frac{N_c}{Z_c + R_c(N_1, \ldots)}$$

Closed models

With the throughput of the classes, we can determine the average number of jobs at each station for all the classes:

$$N_{kc}(N_1, \dots) = X_c(N_1, \dots) \cdot R_{kc}(N_1, \dots)$$

We can then determine the total population at each resource:

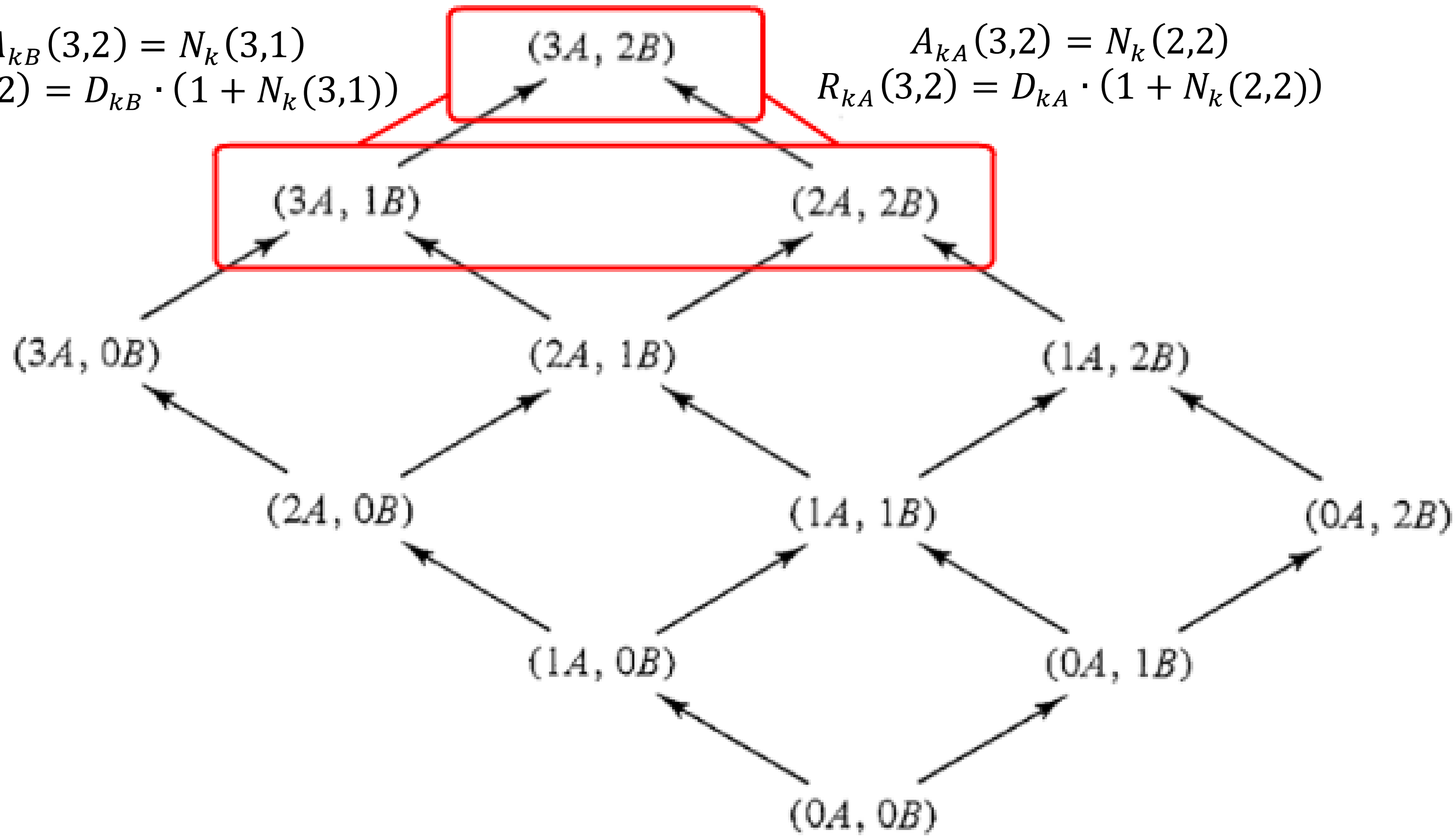
$$N_k(N_1, \dots) = \sum_c N_{kc}(N_1, \dots)$$

Closed models

The main difference with respect to single class models, is that the residence times depend on as many configurations as classes.

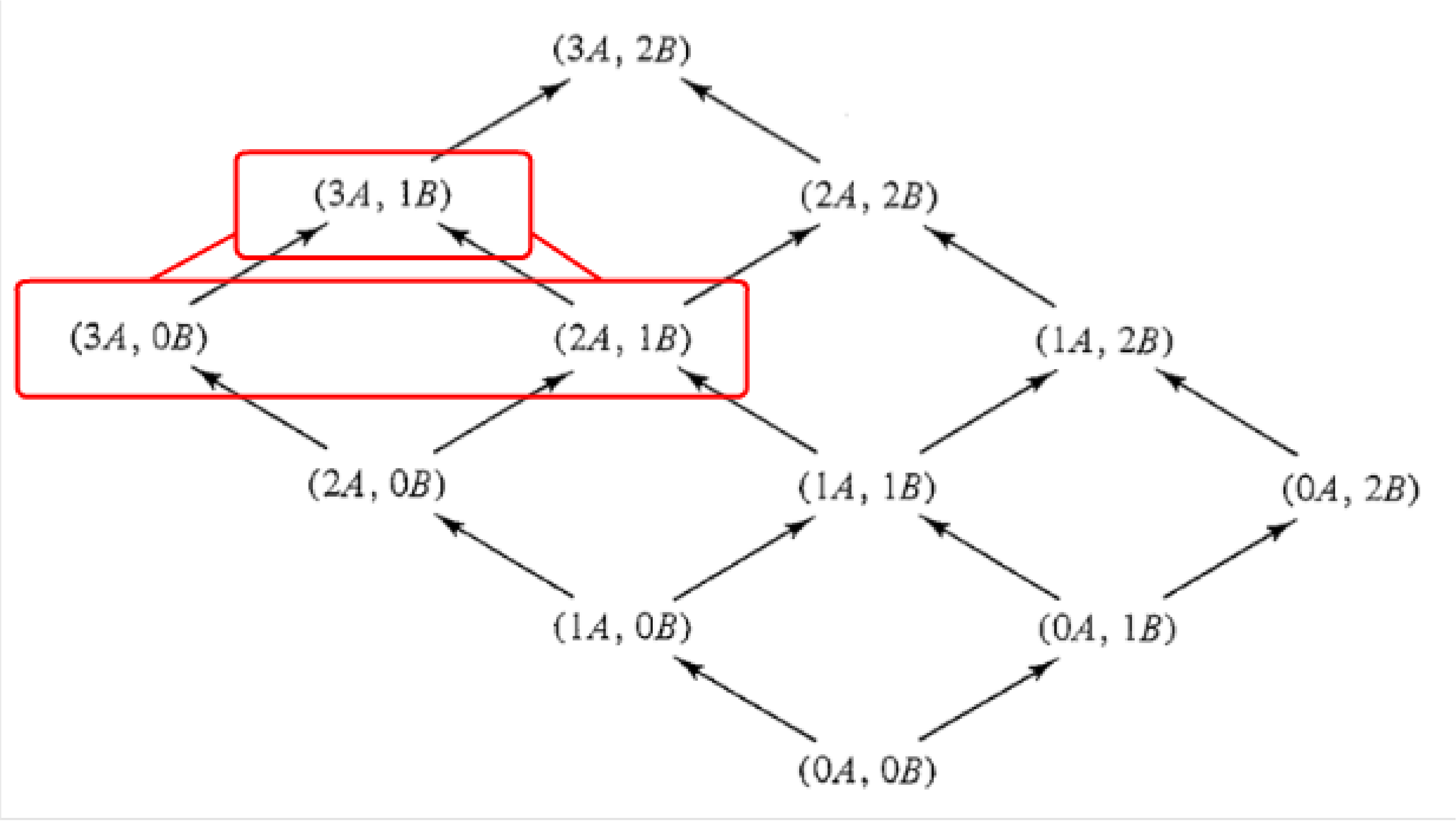
$$A_{kB}(3,2) = N_k(3,1)$$
$$R_{kB}(3,2) = D_{kB} \cdot (1 + N_k(3,1))$$

$$A_{kA}(3,2) = N_k(2,2)$$
$$R_{kA}(3,2) = D_{kA} \cdot (1 + N_k(2,2))$$



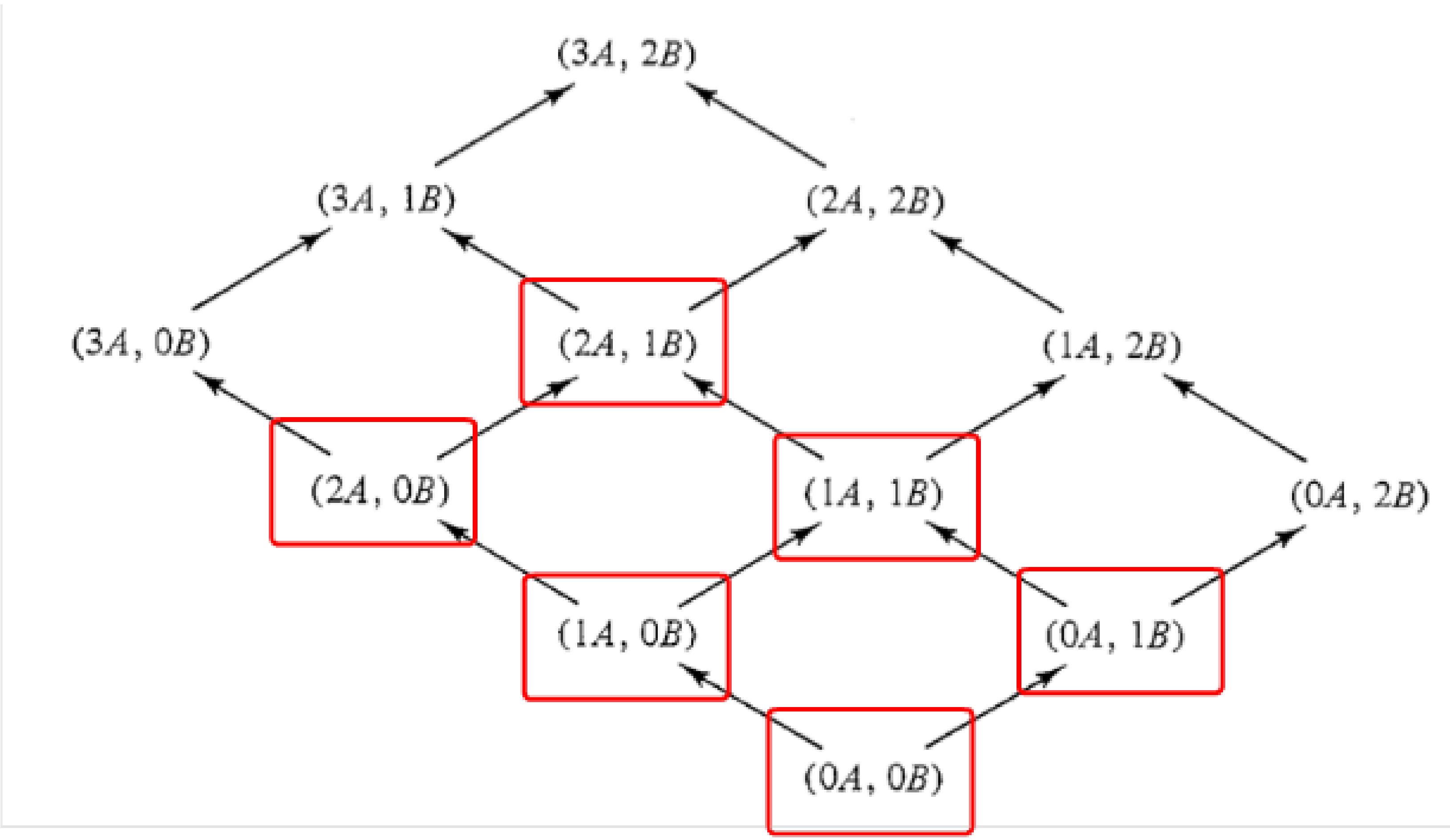
Closed models

Each configuration in turns depends from another set of possible configurations with two less jobs in the system.



Closed models

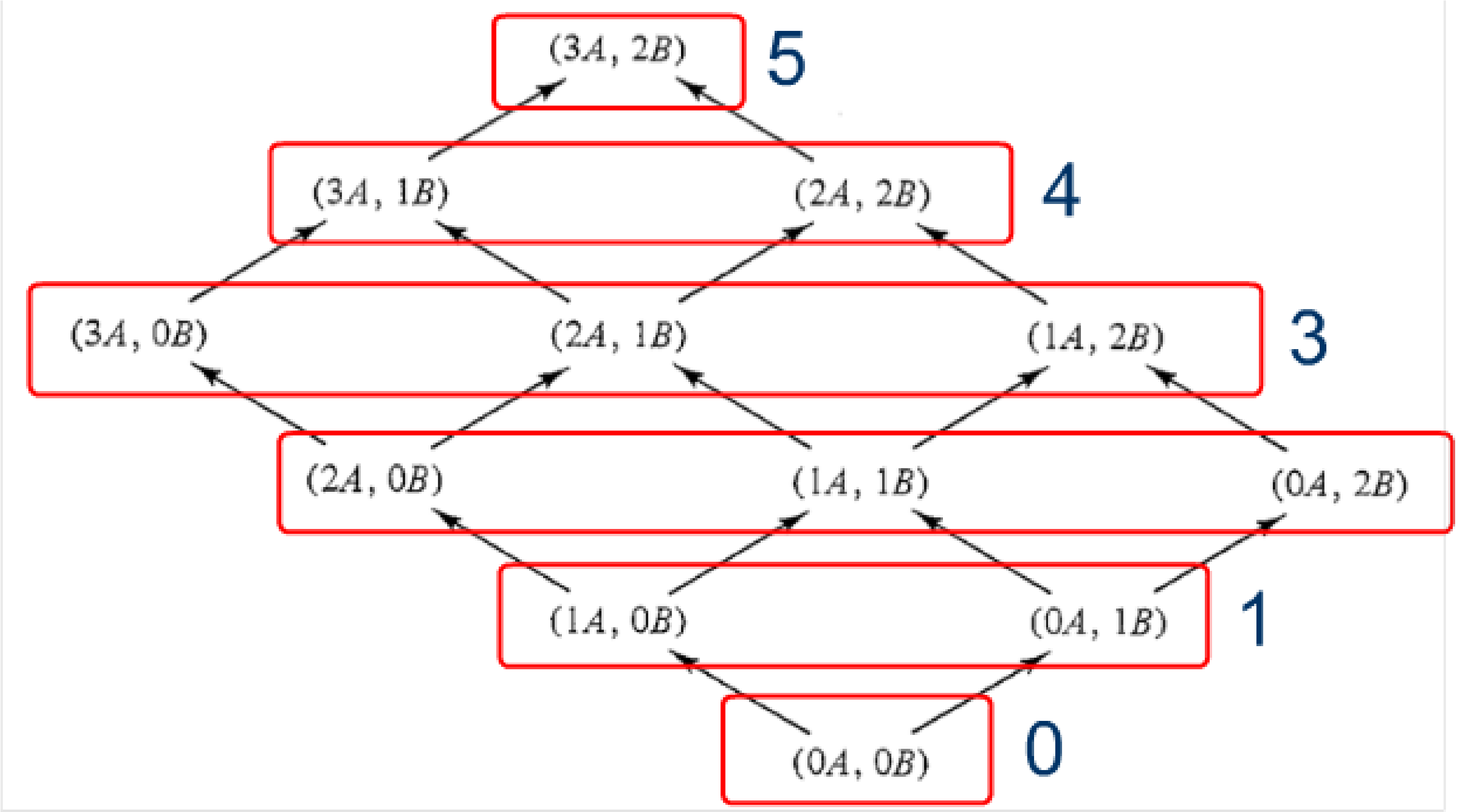
Some configurations however are shared and might be used in several steps.



Closed models

The algorithm starts with an empty system, and adds a jobs at a time to the different classes, maintaining a constant total population.

In this way, the algorithm always has all the information needed to go on with the next iteration.



Closed models

To summarize, we have:

for $k \leftarrow 1$ to K do $Q_k(\vec{0}) \leftarrow 0$

for $n \leftarrow 1$ to $\sum_{c=1}^{C} N_c$ do

for each feasible population $\vec{n} \equiv (n_1, \ldots, n_C)$ with n total customers do

begin

for $c \leftarrow 1$ to C do

for $k \leftarrow 1$ to K do

$$R_{c,k} \leftarrow \begin{cases} D_{c,k} & \text{(delay)} \\ D_{c,k}\left[1 + Q_k(\overrightarrow{n - 1_c})\right] & \text{(queueing)} \end{cases}$$

for $c \leftarrow 1$ to C do $X_c \leftarrow \dfrac{n_c}{Z_c + \sum_{k=1}^{K} R_{c,k}}$

for $k \leftarrow 1$ to K do $Q_k(\vec{n}) \leftarrow \sum_{c=1}^{C} X_c R_{c,k}$

end

This is the hardest step from an algorithmic point of view!

Mixed models

Mixed models can instead be solved using the MVA on the closed classes, after a pre-processing and then applying a post-processing step that accounts for the open classes.

In particular, it happens that the open classes preempt the resources, leaving to the closed classes only the remaining times.

A mixed model is thus stable if it can satisfy all the requests for the considered open classes.

Mixed models

Solution is carried out in three steps:

1. Open classes are considered, determining their utilizations.

2. The demand of the closed classes is *inflated* to account for the time the resources are preempted by jobs from the open classes, and the solution for the inflated closed classes is computed using MVA.

3. The residence times of the open classes are finally computed considering both their utilization and the jobs at the stations belonging to the closed classes.

Mixed models

So, formally:

Determine, for every station k, of every open class c, the utilization U_{kc}.

$$U_{kc} = \lambda_c \cdot D_{kc} \qquad \forall\, k, \forall c \in Op$$

For each station, compute the utilization caused by open classes only U_{kO}.

$$U_{kO} = \sum_{c \in Op} U_{kc} \qquad \forall\, k$$

(Here we use Op to denote the set of open classes)

Mixed models

With the utilizations of the open classes, we can determine an *inflated demand* D'_{kc} for the closed classes:

$$D'_{kc} = \frac{D_{kc}}{1 - U_{kO}} \qquad \forall\, k, \forall c \in Cl$$

Using the MVA with the inflated demands D'_{kc}, we can determine the performances of the closed classes:

$$R_{kc}\,, N_{kc}\,, X_c \qquad \forall\, k, \forall c \in Cl$$

Utilization can be computed with X_c and the initial demands:

$$U_{kc} = X_c \cdot D_{kc} \qquad \forall\, k, \forall c \in Cl$$

(Here we use *Cl* to denote the set of closed classes)

Mixed models

Finally, the average residence time of jobs in the open classes can be computed considering the influences of the closed classes:

$$R_{kc} = \frac{D_{kc} \cdot \left(1 + \sum_{d \in Cl} N_{kd}\right)}{1 - U_{kO}} \qquad \forall\, k, \forall c \in Op$$

The average number of jobs in the open classes is computed using Little's law:

$$N_{kc} = \lambda_{kc} \cdot\ R_{kc} \qquad \forall\, k, \forall c \in Op$$

Mixed models

To summarize:

$$U_{kc} = \lambda_c \cdot D_{kc} \qquad \forall\, k, \forall c \in Op$$

$$D'_{kc} = \frac{D_{kc}}{1 - \sum_{c \in O} U_{kc}} \qquad \forall\, k, \forall c \in Cl$$

Using MVA on D', compute: $R_{kc}\ , N_{kc}\ , X_c \qquad \forall\, k, \forall c \in Cl$

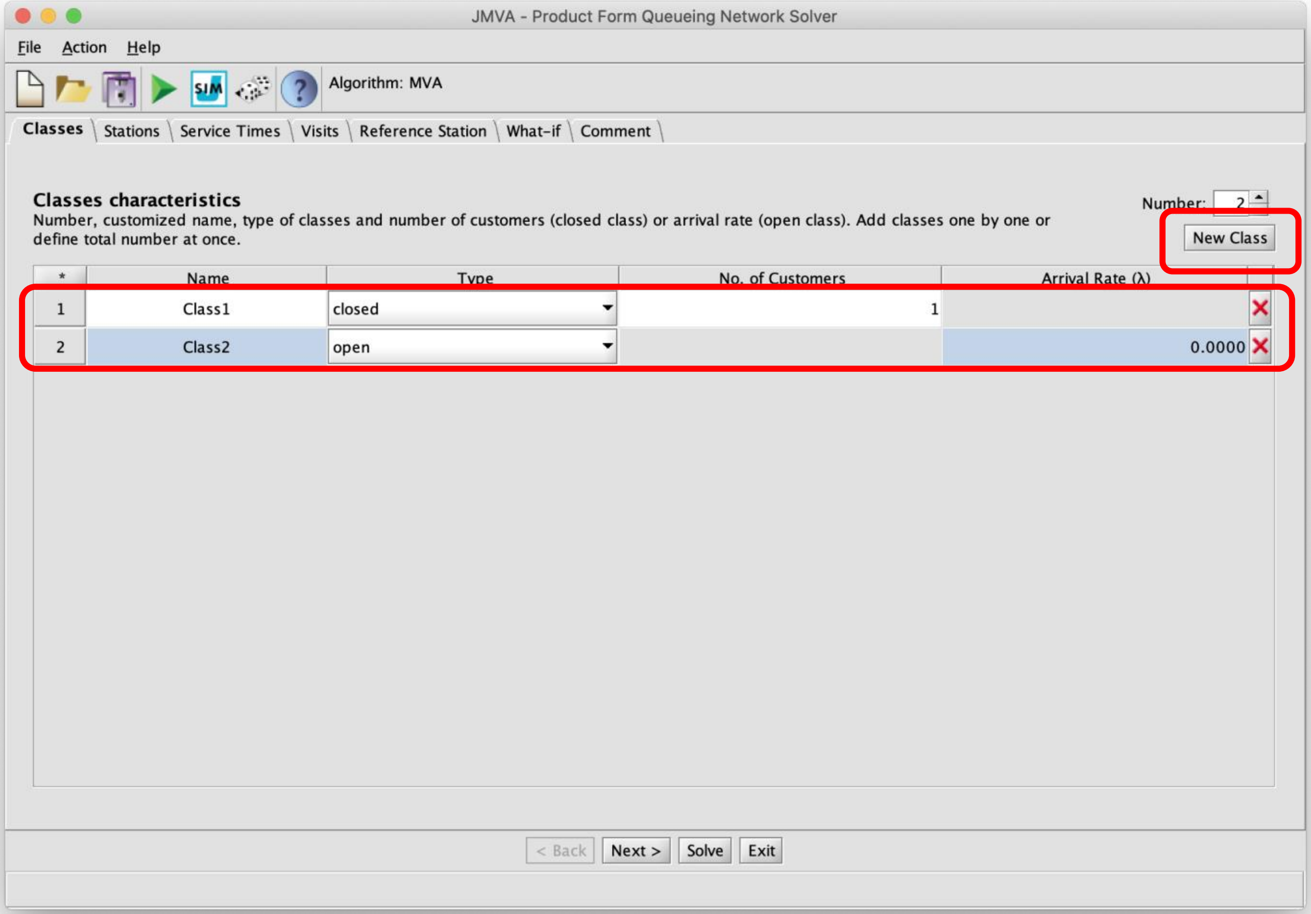
$$U_{kc} = X_c \cdot D_{kc} \qquad \forall\, k, \forall c \in Cl$$

$$R_{kc} = \frac{D_{kc} \cdot (1 + \sum_{d \in Cl} N_{kd})}{1 - U_{kO}} \qquad \forall\, k, \forall c \in Op$$

$$N_{kc} = \lambda_{kc} \cdot\ R_{kc} \qquad \forall\, k, \forall c \in Op$$

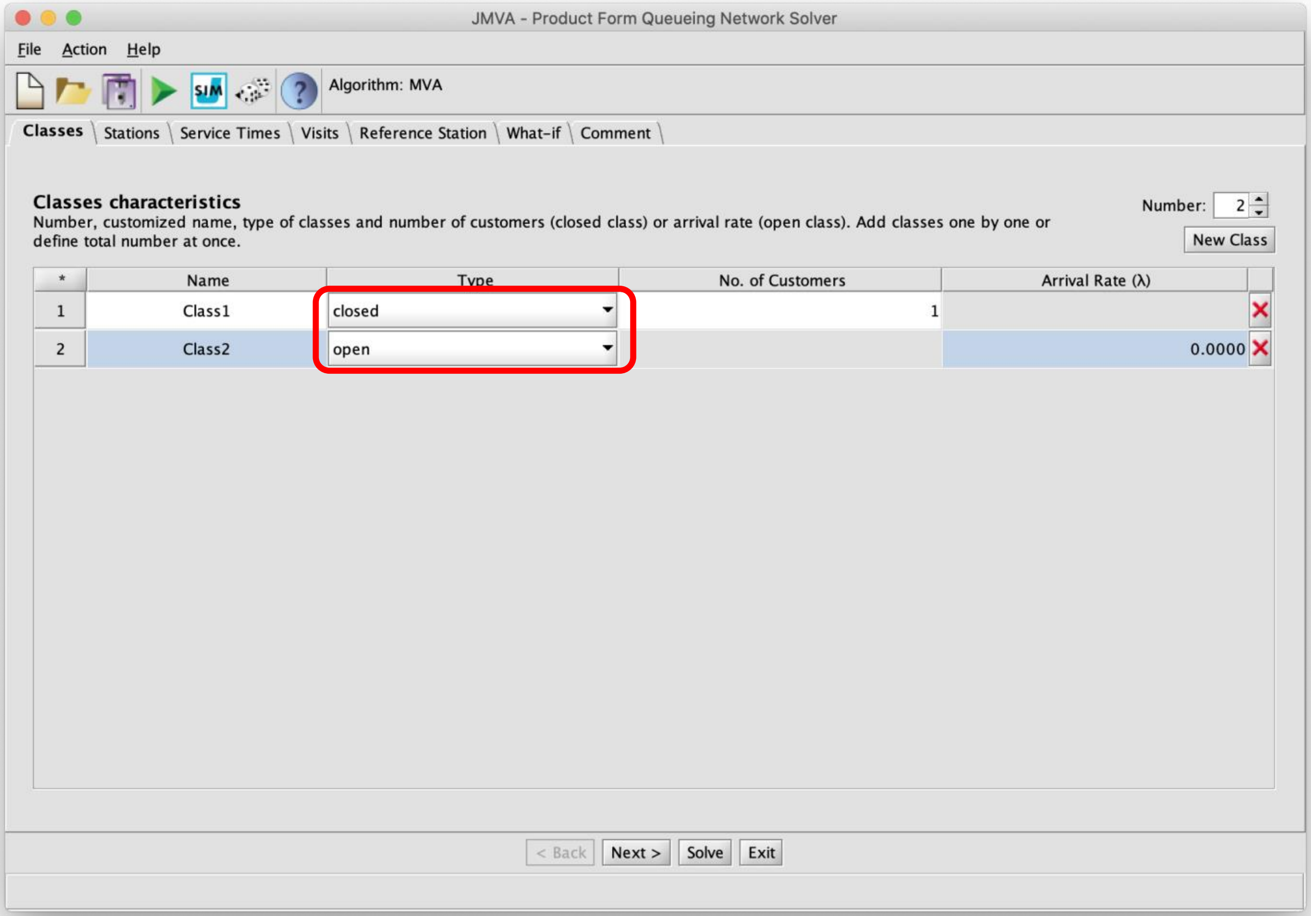
JMVA and multiclass models

The JMVA component of JMT allows to consider multi class models, by adding more classes in its initial page.



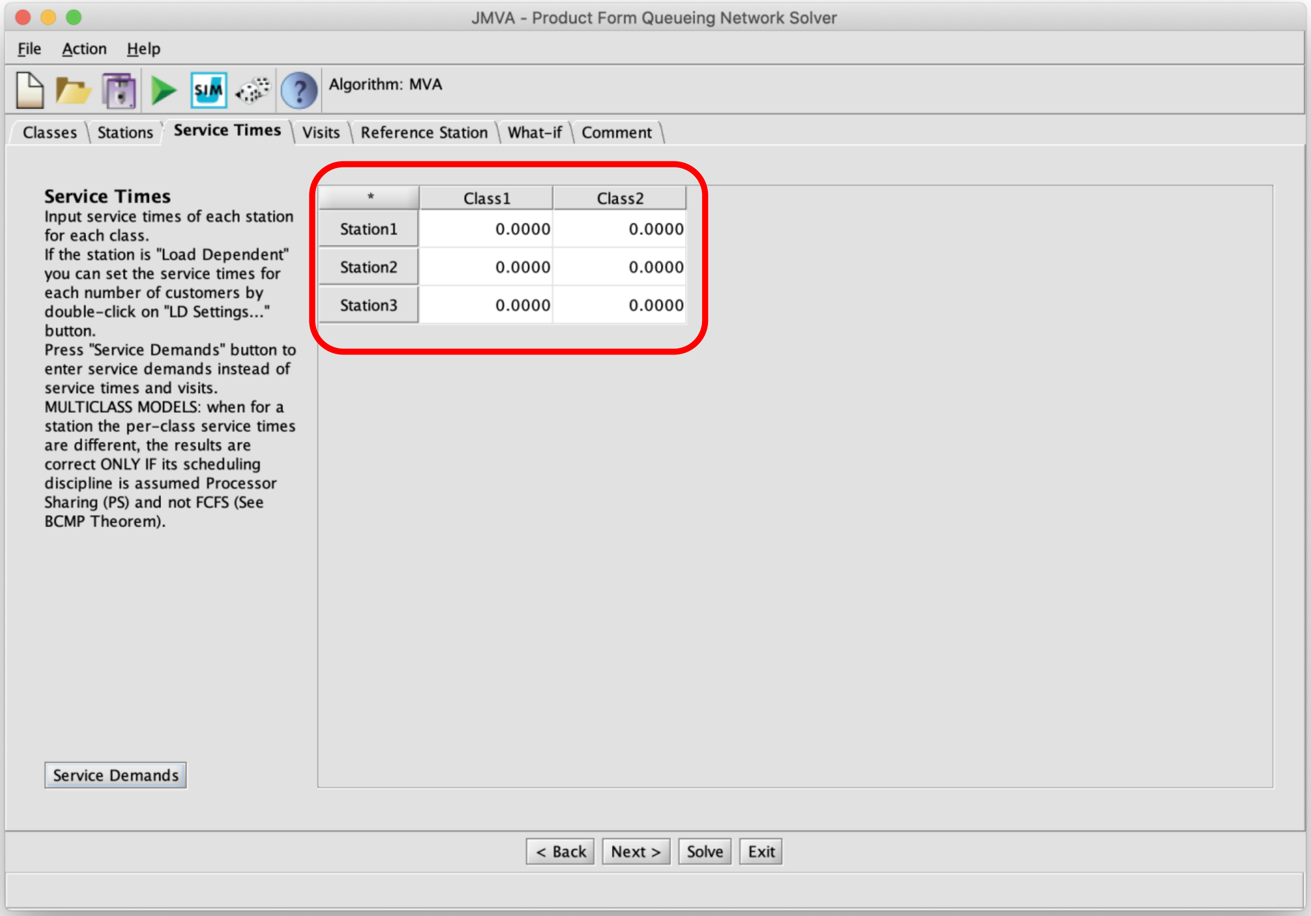
JMVA and multiclass models

Each classes can be either open or closed, allowing thus to define mixed models if needed.



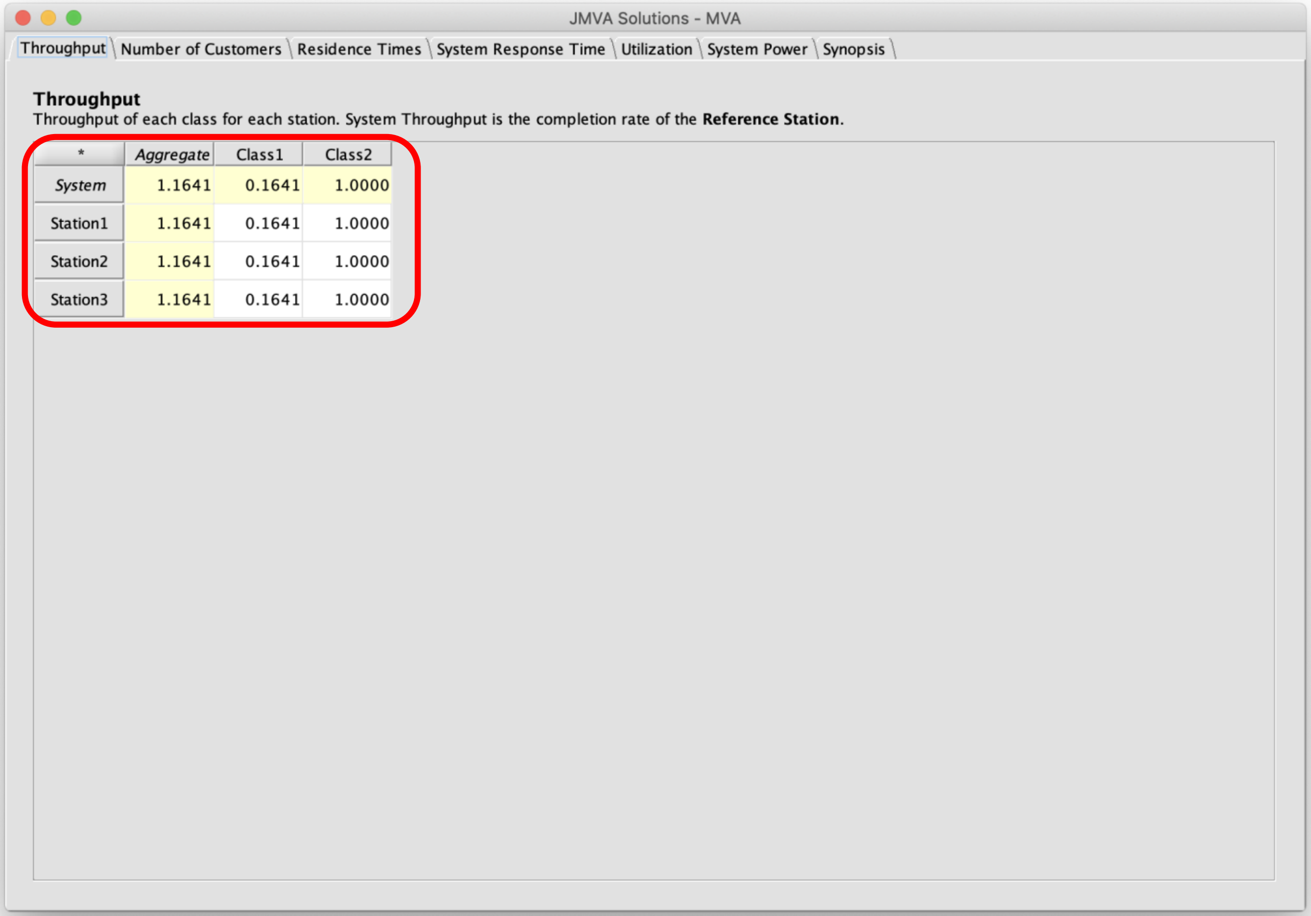
JMVA and multiclass models

Services, visits and demands, as well as reference stations, should be specified per class, in different columns of the table.



JMVA and multiclass models

Performance indices can then be computed per class and per resource. Aggregate measures have a yellow background.

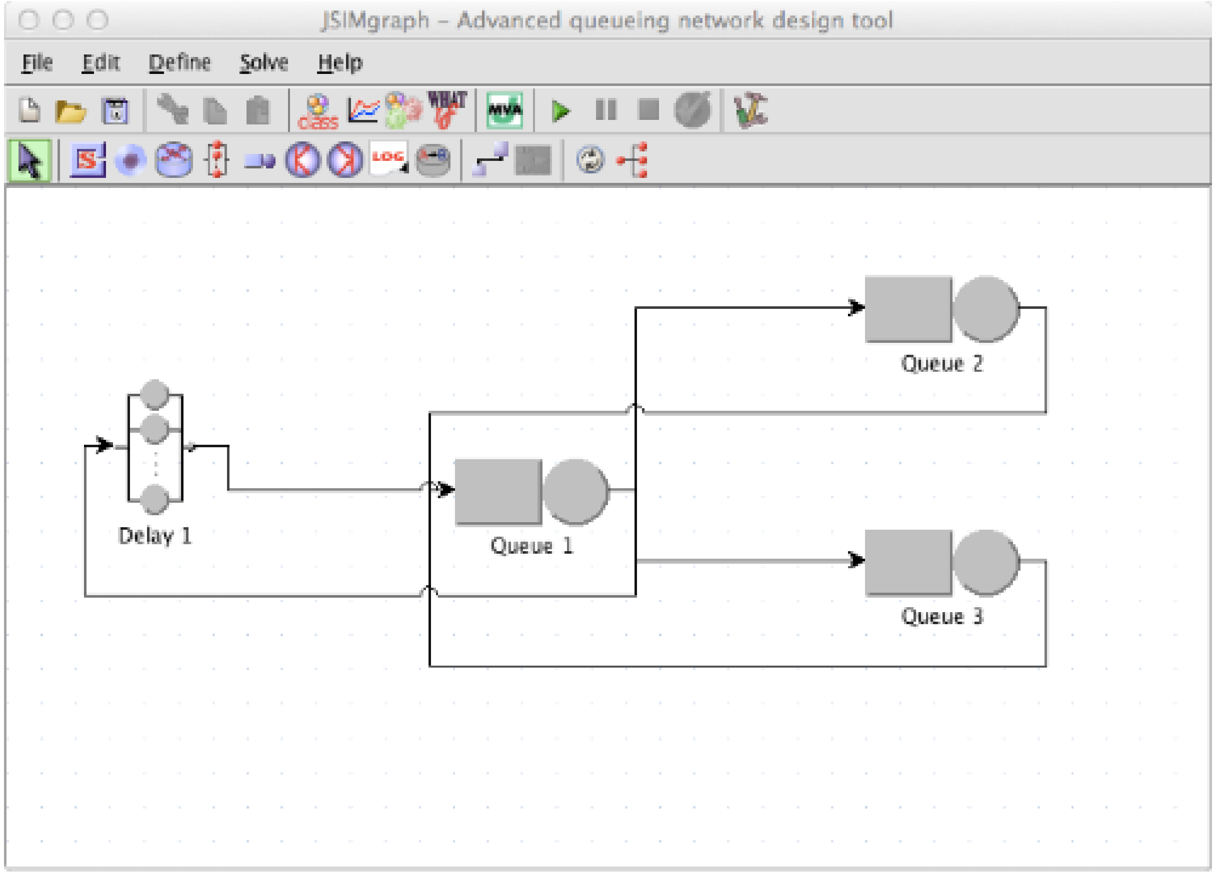


*	*Aggregate*	Class1	Class2
System	1.1641	0.1641	1.0000
Station1	1.1641	0.1641	1.0000
Station2	1.1641	0.1641	1.0000
Station3	1.1641	0.1641	1.0000

Analysis of multi-class models

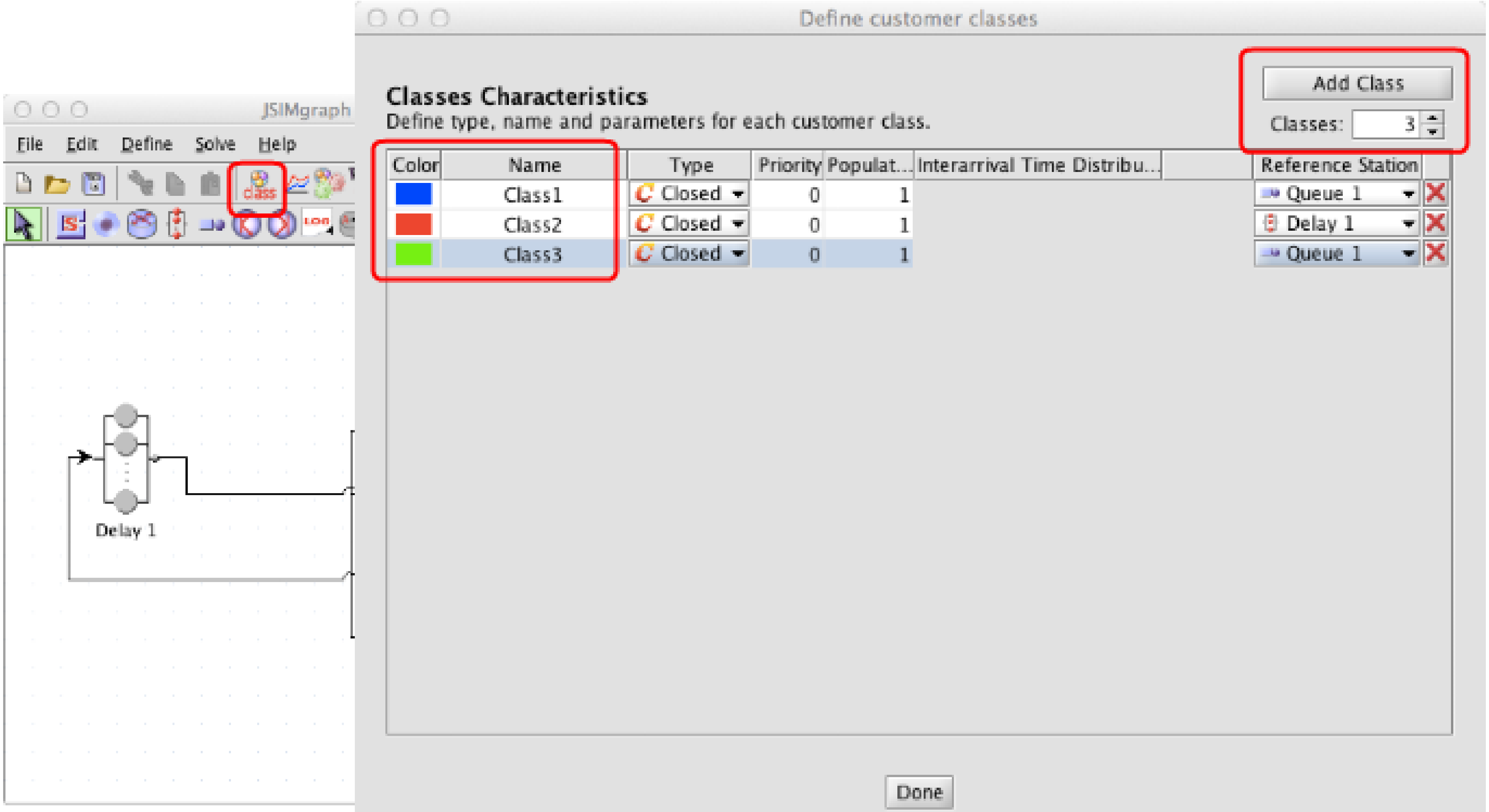
Let's now focus on how multi-class models are handled in JSimGrapg.

The topology of multi-class models in JMT is defined exactly as for single class models.



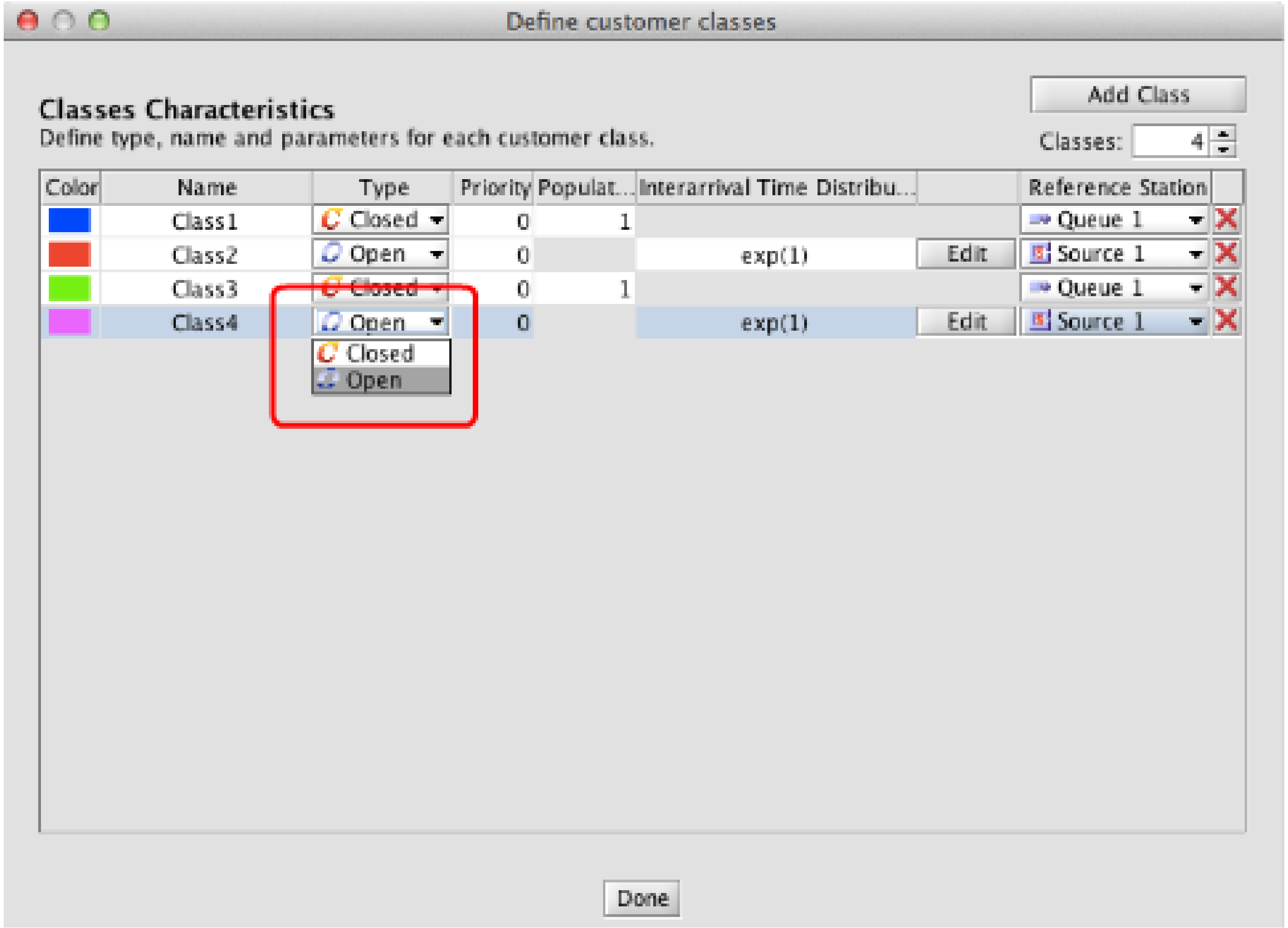
Features of multi-class models

However more than one class is specified in the corresponding page.



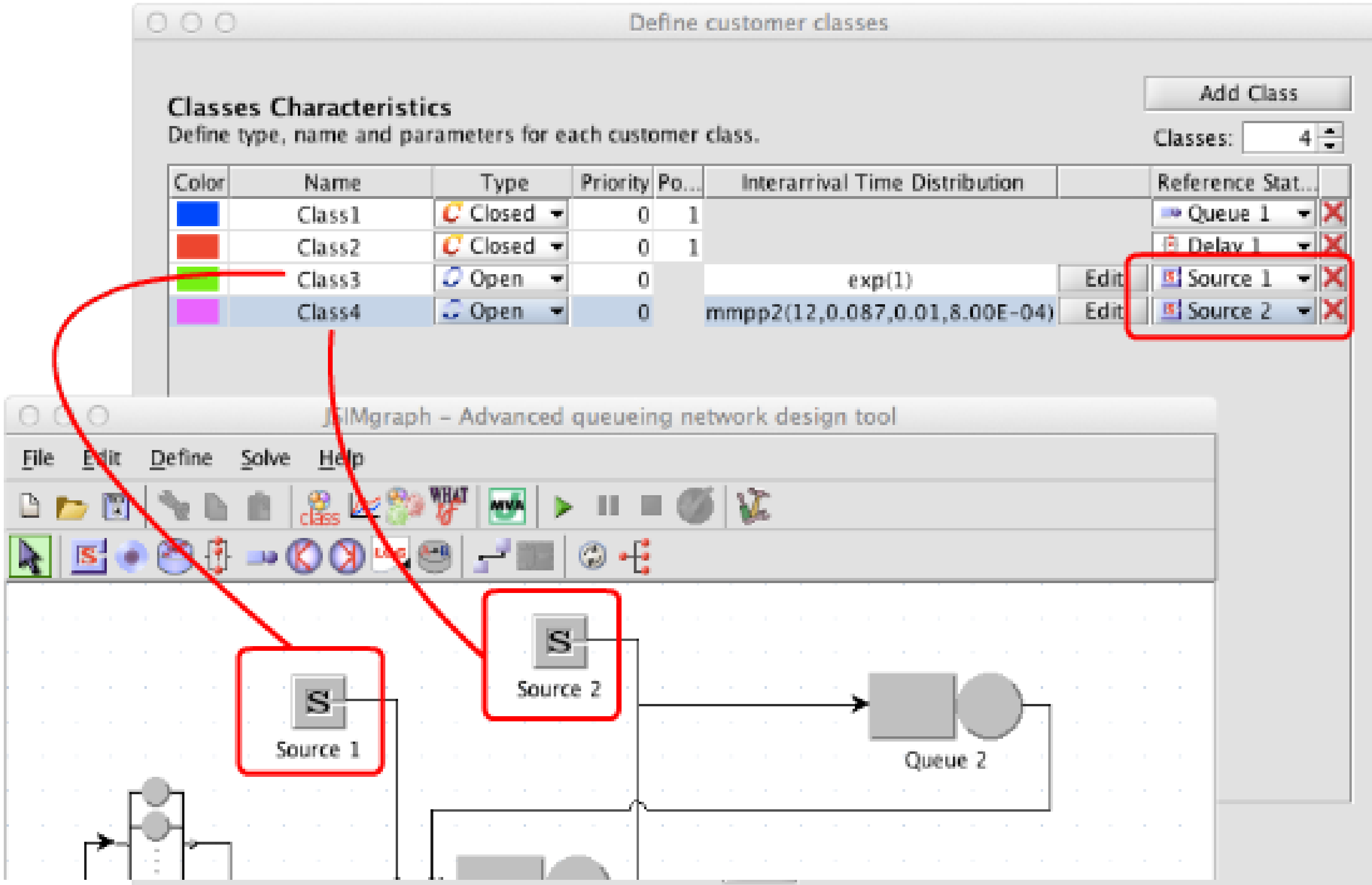
Features of multi-class models

Each class can be defined as either open or closed: using different types of classes allows creating mixed models.



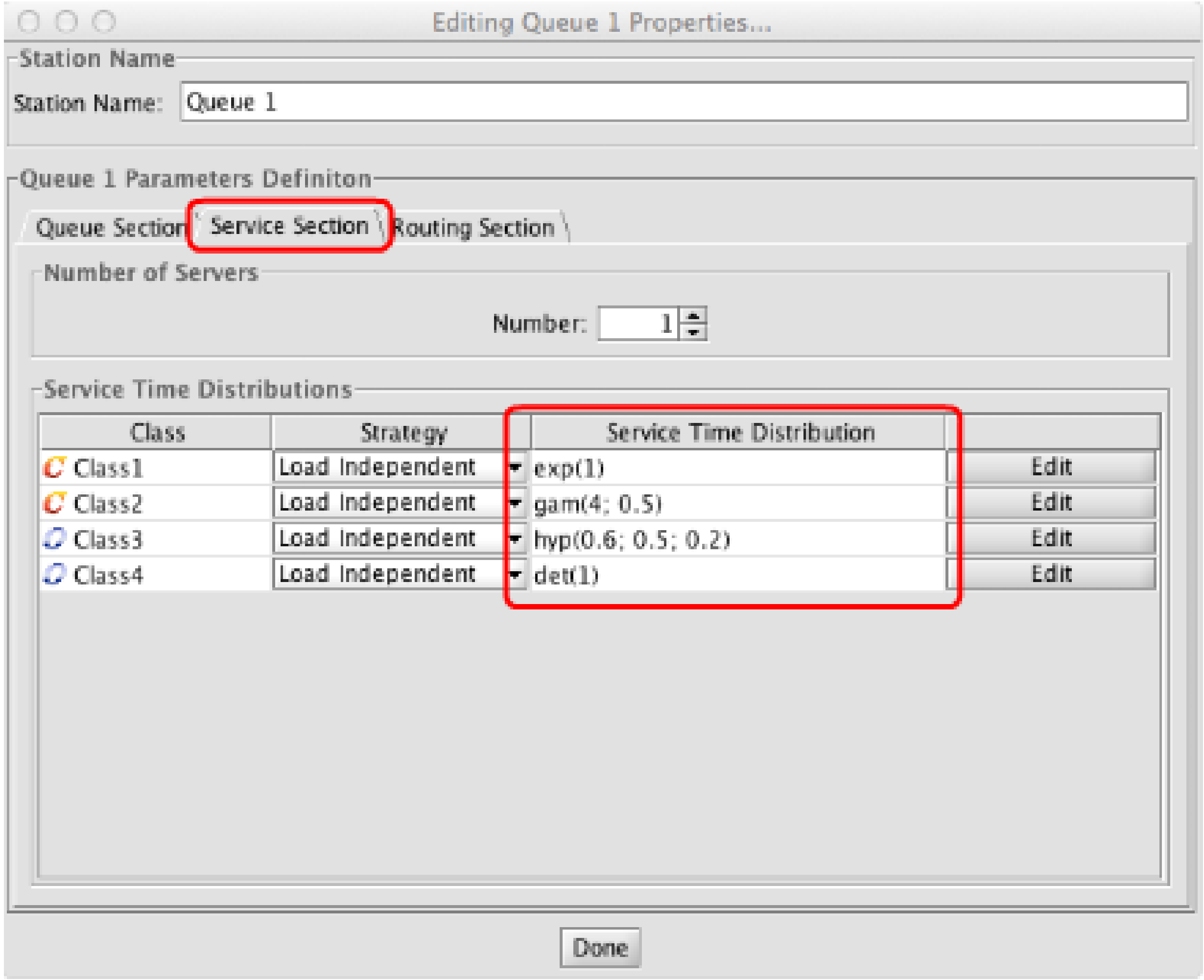
Features of multi-class models

Each open class might have associated a different source node from which the corresponding jobs enter the system.



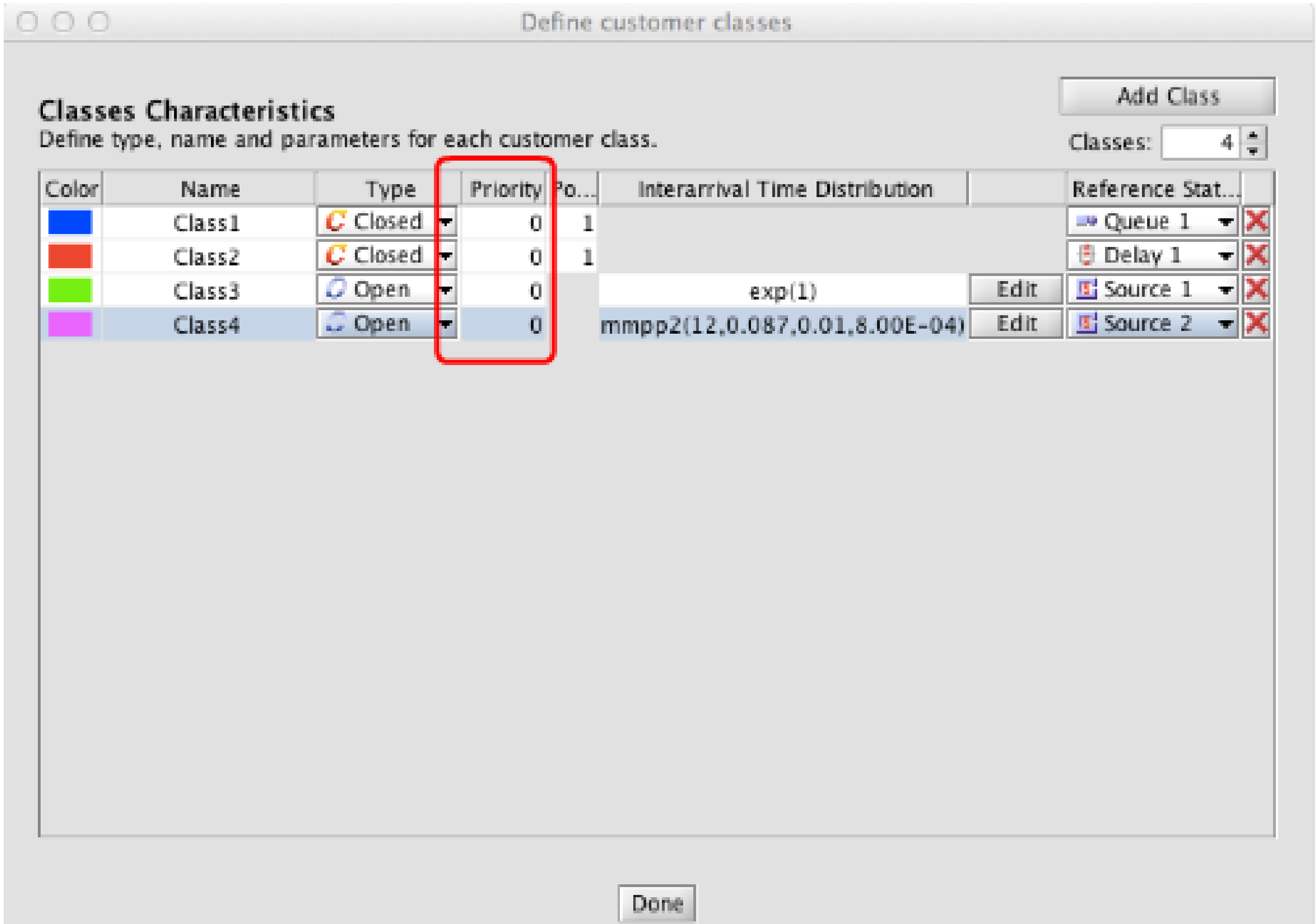
Features of multi-class models

Each queue is characterized by a class dependent service time distribution.



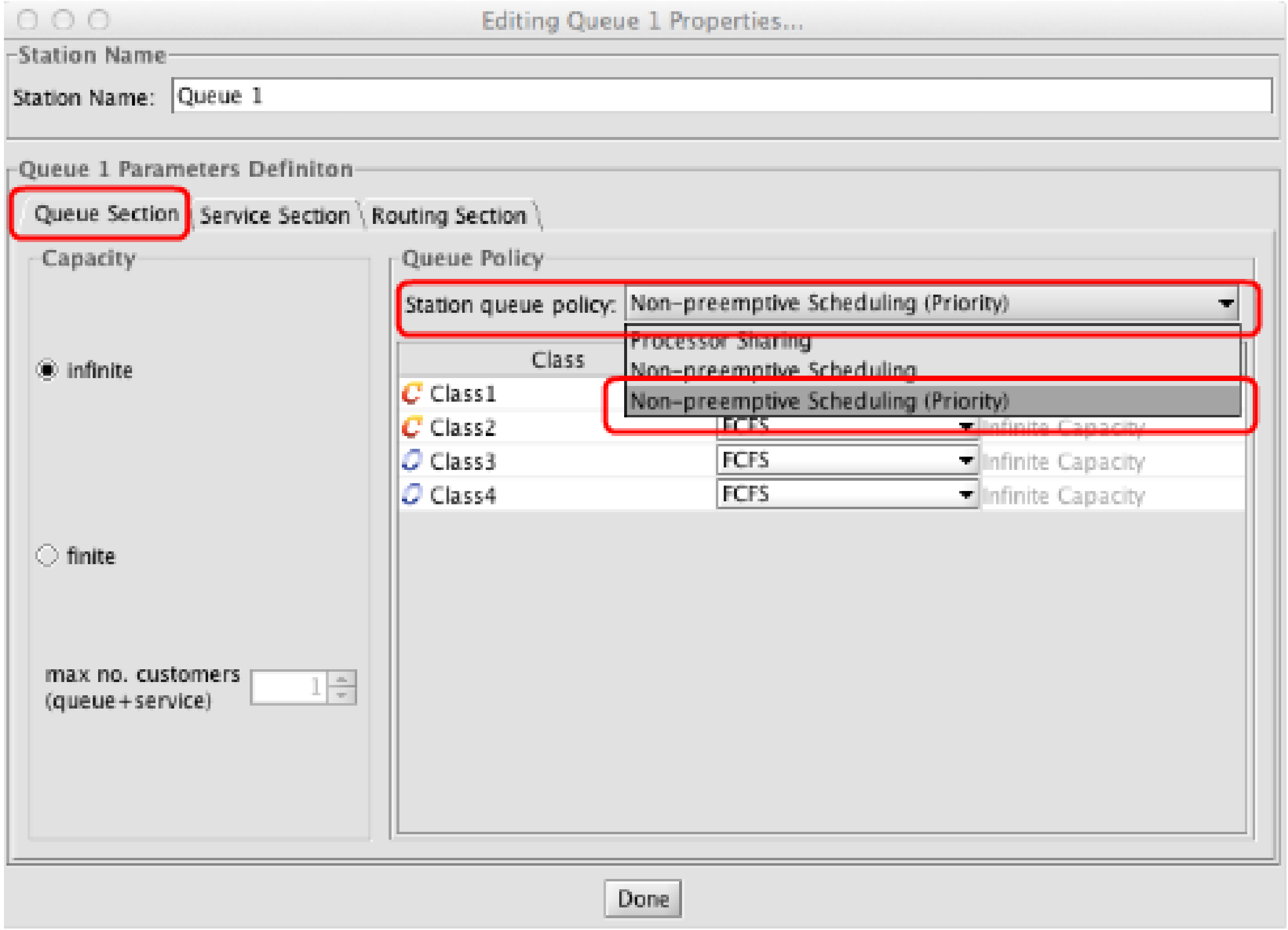
Features of multi-class models

Each class have associated a different priority level in the class definition panel.



Features of multi-class models

Priority can then be used to select jobs in the queue if "*Non-preemptive scheduling (priority)*" is selected in the queue section panel of a node.

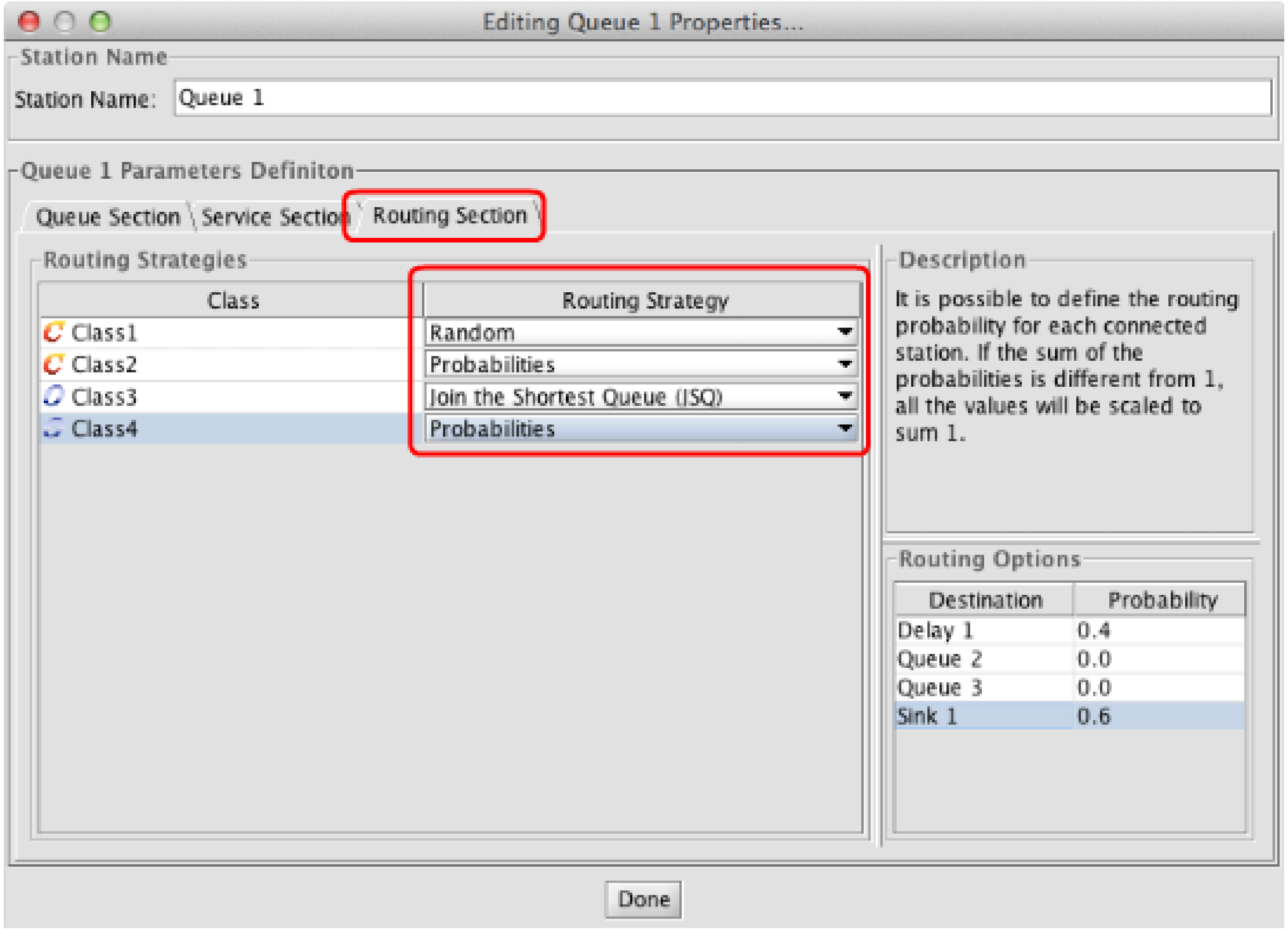


Features of multi-class models

In this case, a job of a class can enter service only if there are no other jobs with a higher priority waiting in the queue.

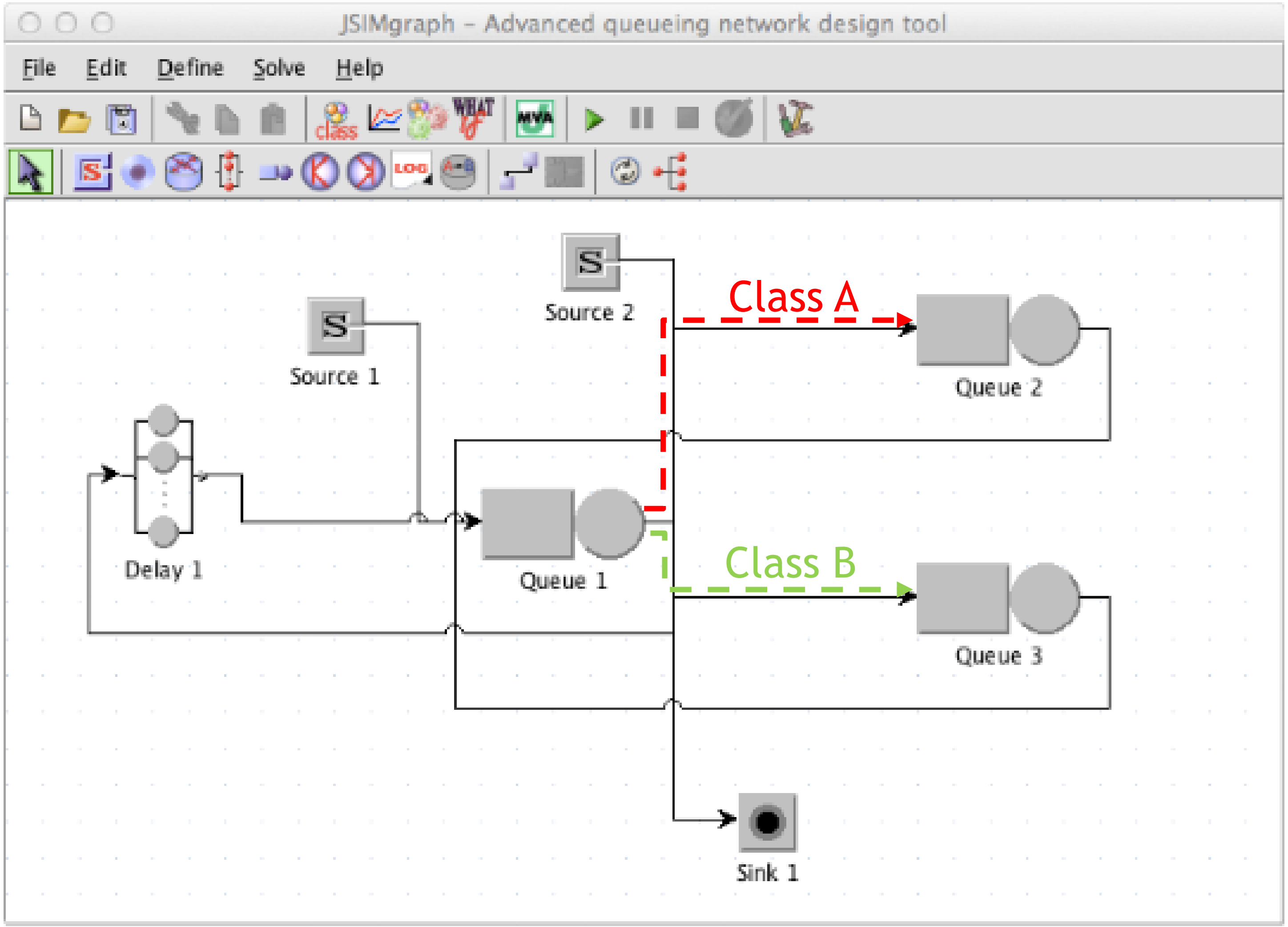
Features of multi-class models

The routing policy specified in the routing section can also be class dependent.



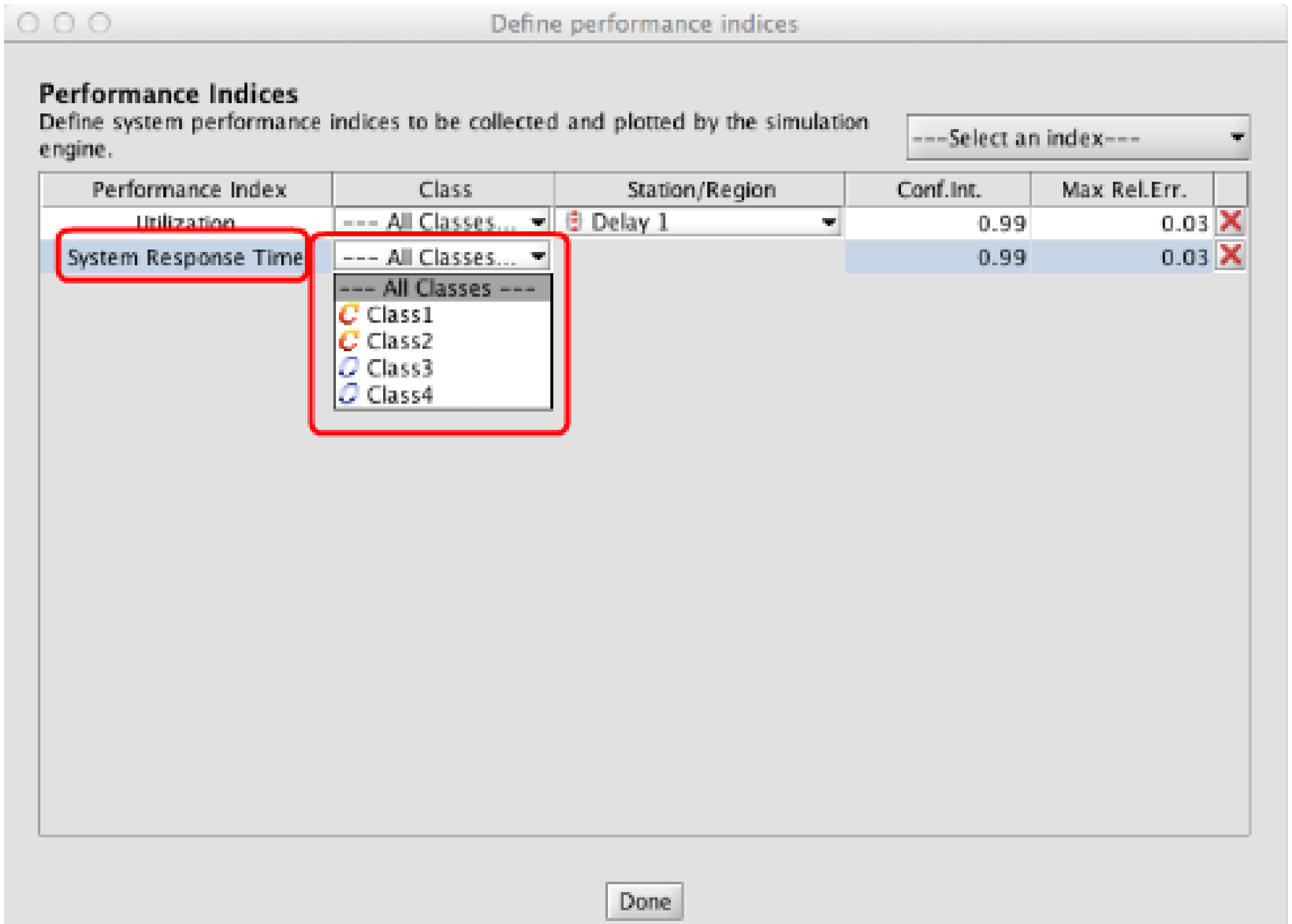
Features of multi-class models

This allows directing jobs of different classes along different routes.



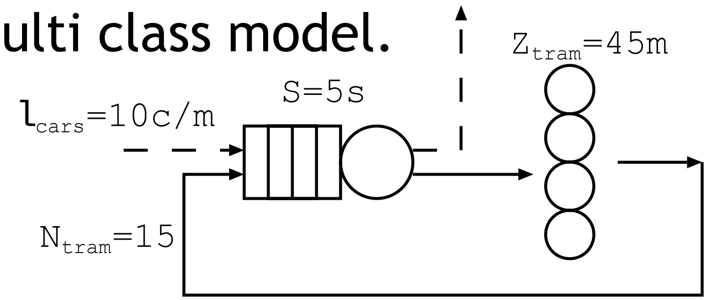
Features of multi-class models

Finally, class dependent performance measures can be defined.



Analysis of Motivating Example

We can analyze the proposed mixed multi class model.



$$U_c = \frac{10 \cdot 5}{60} = 0.833 \quad D' = \frac{5}{1-0.833} = 30s$$

Ns	R	X	N
1	30	0,0003663	0,01098901
2	30,3296703	0,00073251	0,02221686
3	30,6665057	0,00109863	0,03369123
4	31,0107369	0,00146466	0,04542016
5	31,3626048	0,00183059	0,05741201
6	31,7223604	0,00219642	0,06967551
7	32,0902654	0,00256214	0,08221978
8	32,4665934	0,00292776	0,09505432
9	32,8516295	0,00329326	0,10818907
10	33,245672	0,00365865	0,12163441
11	33,6490322	0,00402393	0,1354012
12	34,0620359	0,00438907	0,14950079
13	34,4850237	0,00475409	0,16394506
14	34,9183518	0,00511898	0,17874644
15	35,3623933	0,00548373	0,19391796

$X_T = \lambda_T = 0.005484$ tram / sec

We need to compute the tram inter-arrival time, which is $1/\lambda_T$

$$T_T = \frac{1}{0.005484} 60 = 3.04\, min$$

(this is a very rough approximation, since it considers that the tram will not be slowed down outside the considered street segment)