POLITECNICO DI MILANO

Performance Evaluation and Applications





Closed models

Motivating example

An ***automated warehouse*** has 3 robots that takes the goods from the shelves to the delivery area. Each robot has its own charging station, that requires a total of 30min to fully recharge its battery. Only one robot at a time can operate either in the shelves or delivery area, requiring respectively 6min and 4min to complete their tasks. Travelling between the zones can be done in parallel, and takes an average of 2min. Each robot has to return to the charging station on the average every 25 deliveries.

Management is interested in determining:

1. Average time between recharges for each robot.
2. System throughput (how many goods are delivered per hour).

Reference station

The computation of visits in closed model is more complex than in open models: as in open models, jobs can visit the same station several times, or might skip some stations.

Completions at each station can be counted as in open models, and visits are defined as:

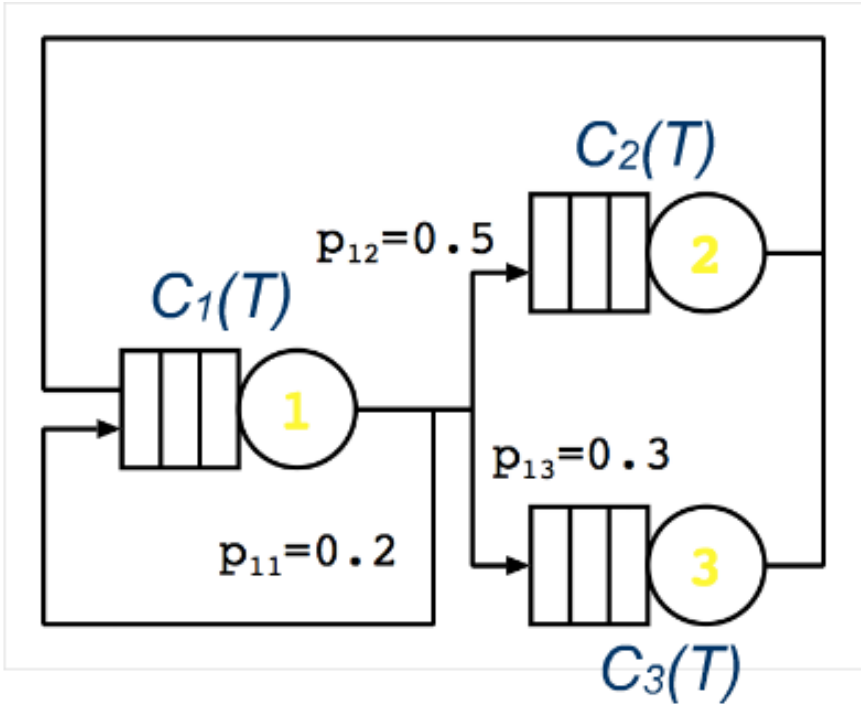
$$v_k = \lim_{T \to 0} \frac{C_k(T)}{C(T)}$$

Reference station

However, as opposed to open models, it is not easy to define the count of jobs that have completed their execution *C(T)*.

$$v_k = \lim_{T \to 0} \frac{C_k(T)}{C(T)}$$

$C(T) = ?$



Reference station

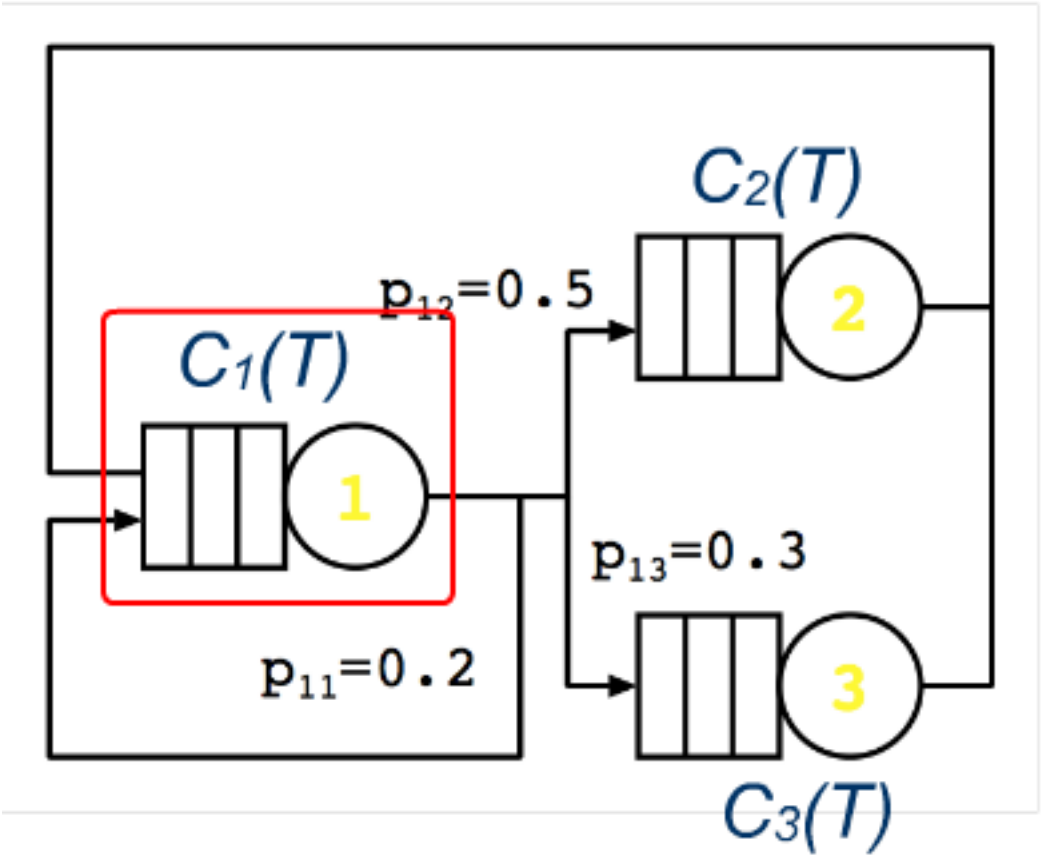
In order to compute the visits and the system throughput, we must be able to determine when a job has finished its service.

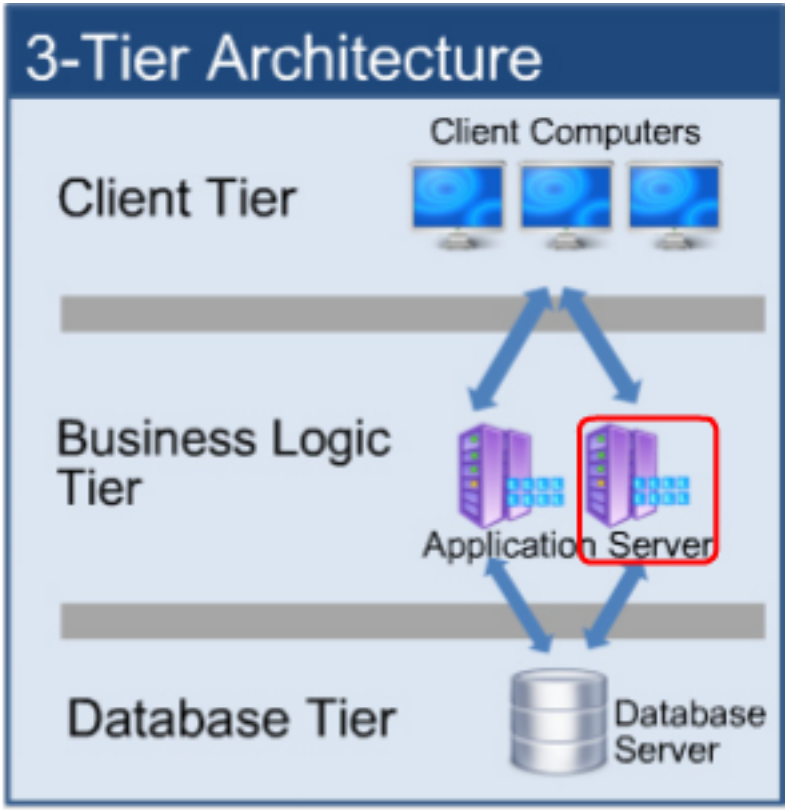
We *define* that a job has ended its service whenever it exits from a specific station, and we address this special station as the *Reference Station*.

The definition of the reference station thus becomes an important part of the model specification.

Reference station

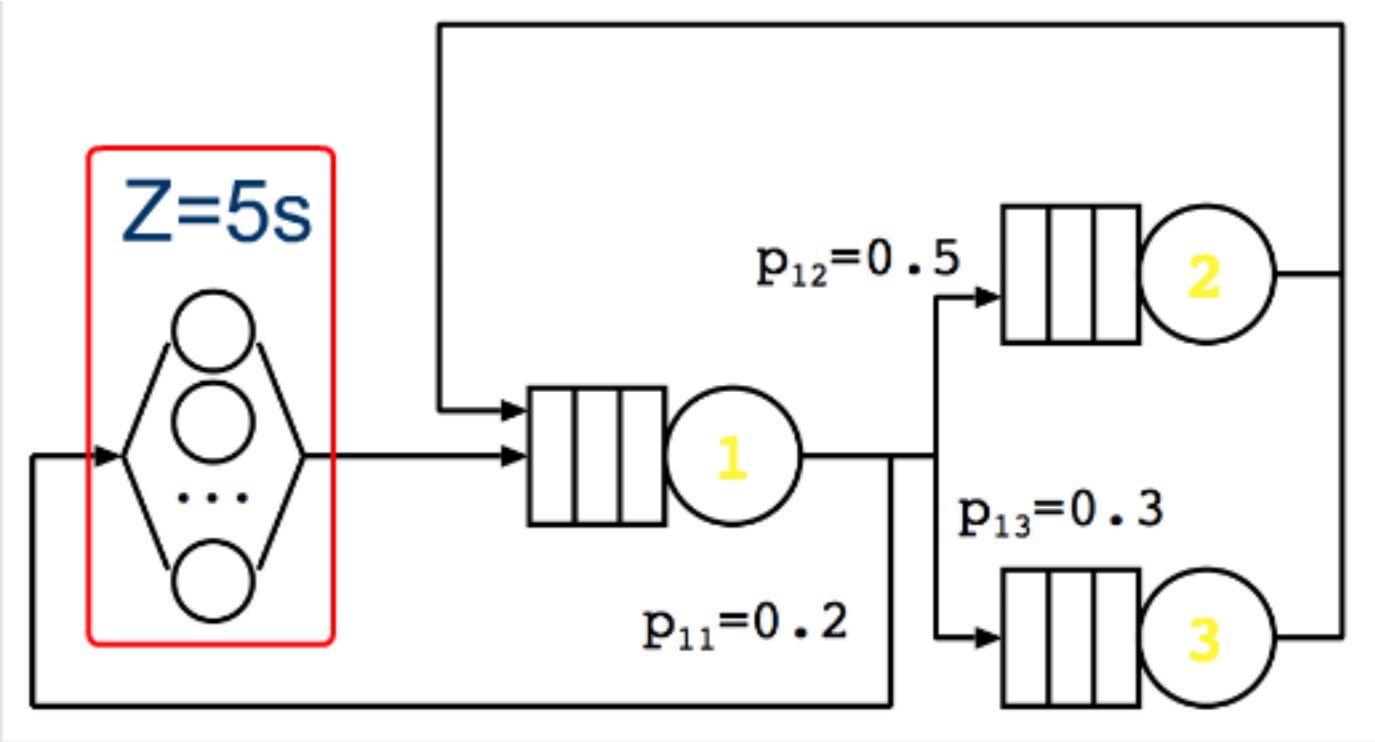
In a three-tier batch application, the reference station can be for example the web server.





Reference station

In a time-sharing systems, the reference station is usually the infinite server that defines the think time of the users.



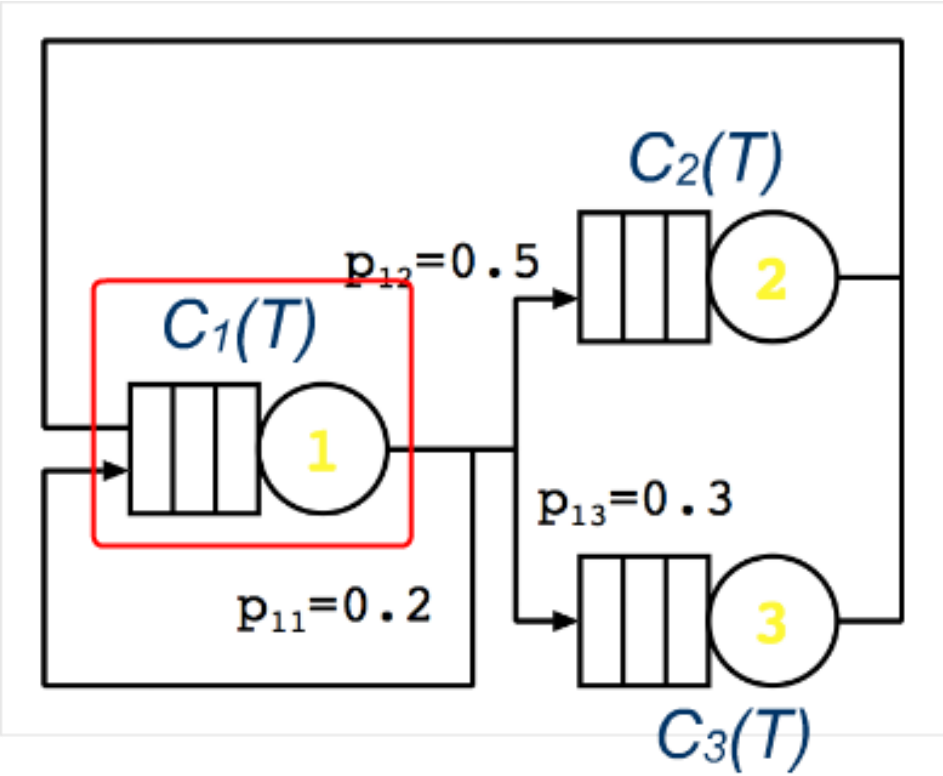
Reference station

The total number of completion of jobs $C(T)$ is then defined as the number of completions at the reference station $C_{ref}(T)$.

In this way, the system throughput corresponds to the throughput of the reference station, that is $X = X_{ref}$.

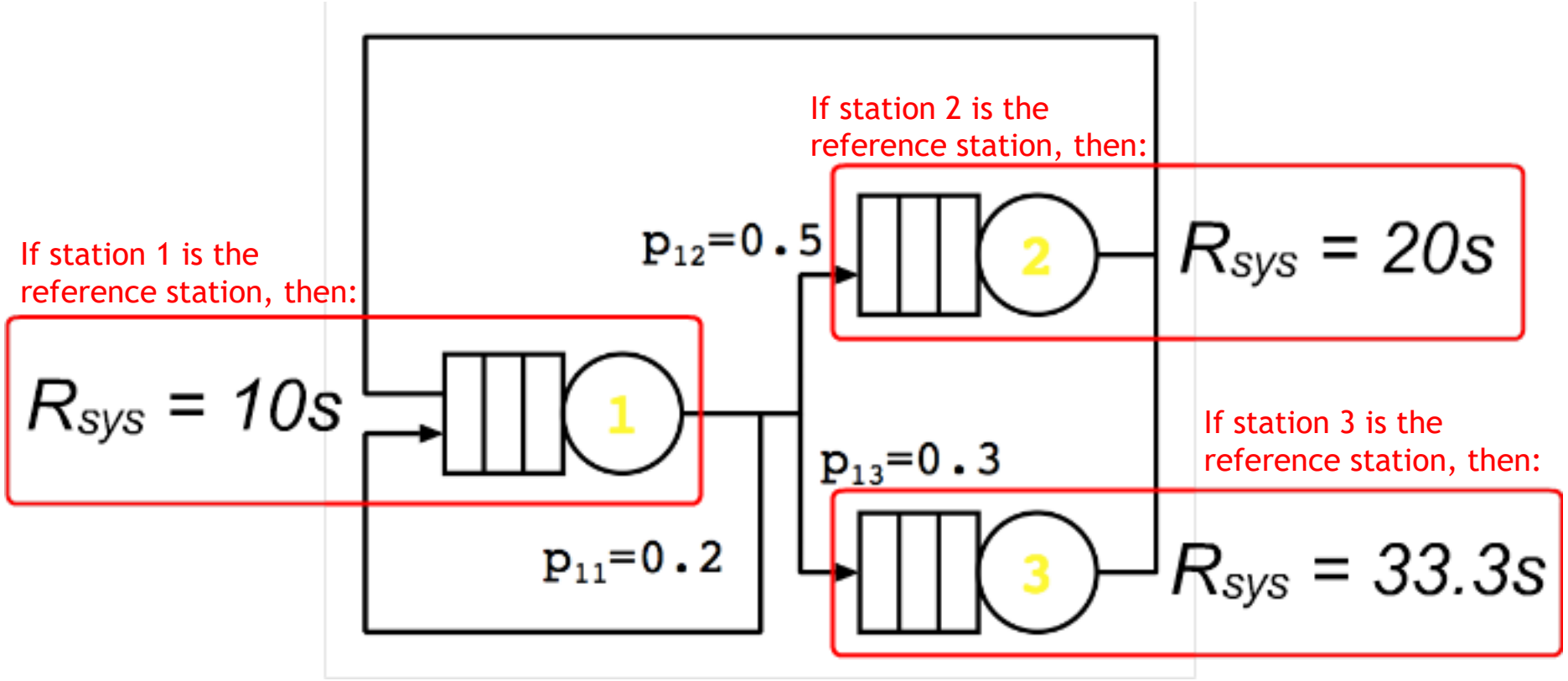
$$C(T) = C_1(T)$$

$$X = X_1$$



Reference station

Note that the system can have very different performances by changing the reference station, in terms of throughput and, as we will see later, system response time.



Computing visits in closed models

Visits in closed models can be computed in the same way as in open models.

However, in this case there are no input terms, and visits are computed against the arrival rate (the throughput) of the reference station λ_{ref}.

$$\begin{cases} \lambda_k = \sum_{i=1}^{K} \lambda_i \cdot p_{ik} \\ \dots \end{cases}$$

$$v_k = \frac{\lambda_k}{\lambda_{ref}}$$

Computing visits in closed models

As we did for open models, we can directly compute the visits dividing with the system arrival rate, which in this case corresponds to the one of the reference station:

$$\begin{cases} \lambda_k = \sum_{i=1}^{K} \lambda_i \cdot p_{ik} \\ \dots \end{cases}$$

$$\begin{cases} \frac{\lambda_k}{\lambda_{ref}} = \sum_{i=1}^{K} \frac{\lambda_i}{\lambda_{ref}} \cdot p_{ik} \\ \dots \end{cases}$$

$$v_k = \frac{\lambda_k}{\lambda_{ref}}$$

$$\begin{cases} v_k = \sum_{i=1}^{K} v_i \cdot p_{ik} \\ \dots \end{cases}$$

Computing visits in closed models

Without extra care, the system of equation would have an infinite number of solutions, since the rows are not linearly independent.

The wanted solution is then obtained by setting the visits to the reference station equal to one.

$$v_{ref} = 1$$

Computing visits in closed models

Visits can then be computed with the following equations:

$$\begin{cases} v_k = \sum_{i=1}^{K} v_i \cdot p_{ik} & \forall k \neq ref \\ v_{ref} = 1 \end{cases}$$

Computing visits in closed models

In matrix form, we define vector ***l*** as vector that has term corresponding to the reference station equal to one, and all the others equal to zero.

We then define matrix $\boldsymbol{P_0}$ as matrix P in which we have replaced the column corresponding to the reference station, with vector composed of all zeros. This is because the visits to the reference station do not depend on the inputs, but they are fixed to one.

$$\mathbf{l}= \left| l_k : \begin{cases} 1 & k = ref \\ 0 & k \neq ref \end{cases} \right| \quad \boldsymbol{v} = |\cdots v_k \cdots| \quad \boldsymbol{P}_0 = \left| p'_{ij} : \begin{cases} 0 & j = ref \\ p_{ij} & j \neq ref \end{cases} \right|$$

$$\boldsymbol{v} = \boldsymbol{l} \cdot (\boldsymbol{I} - \boldsymbol{P}_0)^{-1}$$

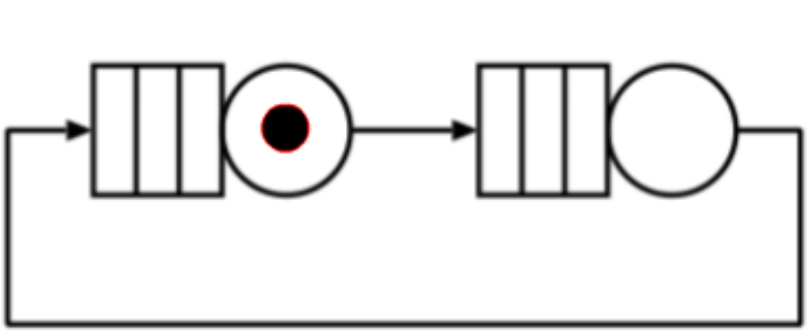
Analysis of closed models

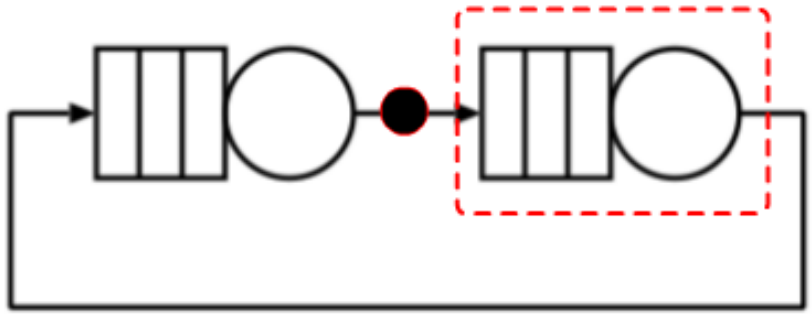
As for separable open queueing network models, the solution of closed ones is based on the computation of $A_k(N)$, the average number of jobs at the arrival.

However, the expression of $A_k(N)$ in *closed models* is different from the one used in open models.

If we consider a network with two stations, identical demand and a single job, we can see immediately that the average number of jobs is equal to one half: $N_k(N) = 0.5$.

However, since there is just one job, the number of jobs that are found when the costumer enters a station is always $A_k(1) = 0$.





Analysis of closed models

In closed models we can use the *"Arrival Theorem"* which states *that the number of jobs that a costumer finds in the queue at its arrival, is equal to the average queue length of that station when system has one less job.*

$$A_k(N) = N_k(N-1)$$

Mean Value Analysis (MVA)

The performance indices can then be computed in an iterative way, starting from an empty system and adding one job per iteration.

This technique is called "*Mean Value Analysis*" (*MVA*).

Let us imagine that the system has been studied up to a workload of *N-1* jobs. Using the arrival theorem we can determine the residence time at each station when the population increases of one job (*N* jobs).

$$\mathrm{R}_k(N) = (1 + A_k(N)) \cdot D_k = \big(1 + N_k(N-1)\big) \cdot D_k$$

Mean Value Analysis (MVA)

Summing up the residence time at all the stations, we can determine the *system response time*.

$$R(N) = \sum_k \mathrm{R}_k(N)$$

Inverting the *response time law*, the system throughput can be determined.

$$R(N) = \frac{N}{X(N)} - Z$$

$$\Downarrow$$

$$X(N) = \frac{N}{R(N) + Z}$$

Mean Value Analysis (MVA)

Using *Little's law*, the queue length of each station can finally be determined.

$$N_k(N) = X(N) \cdot R_k(N)$$

The algorithm can then increase the population to *N+1* jobs, since for the arrival theorem we have that $A_k(N+1) = N_k(N)$, and the process can be repeated.

$$A_k(N+1) = N_k(N)$$

$$\mathrm{R}_k(N+1) = \big(1 + A_k(N+1)\big) \cdot D_k = \big(1 + N_k(N)\big) \cdot D_k$$

Mean Value Analysis (MVA)

The starting point corresponds to an empty system.

$$N_k(0) = 0$$

$$A_k(1) = N_k(0) = 0$$

$$\mathrm{R}_k(1) = \big(1 + A_k(1)\big) \cdot D_k = D_k$$

Mean Value Analysis (MVA)

To summarize we have:

for $k \leftarrow 1$ to K do $Q_k \leftarrow 0$

for $n \leftarrow 1$ to N do

begin

for $k \leftarrow 1$ to K do $R_k \leftarrow \begin{cases} D_k & \text{(delay centers)} \\ D_k(1 + Q_k) & \text{(queueing centers)} \end{cases}$

$$X \leftarrow \frac{n}{Z + \sum_{k=1}^{K} R_k}$$

for $k \leftarrow 1$ to K do $Q_k \leftarrow XR_k$

end

Mean Value Analysis complexity

The mean value analysis computes the solution of a model with complexity $O(N \cdot K)$.

However, as a by-product, it computes the solution for the models with population sizes *1* to *N-1*.

This can be particularly effective when performing sizing studies, where the evolution of the performance against the population is required.

The response time law

In time-sharing systems, the "think time" is usually not considered in the system response time.

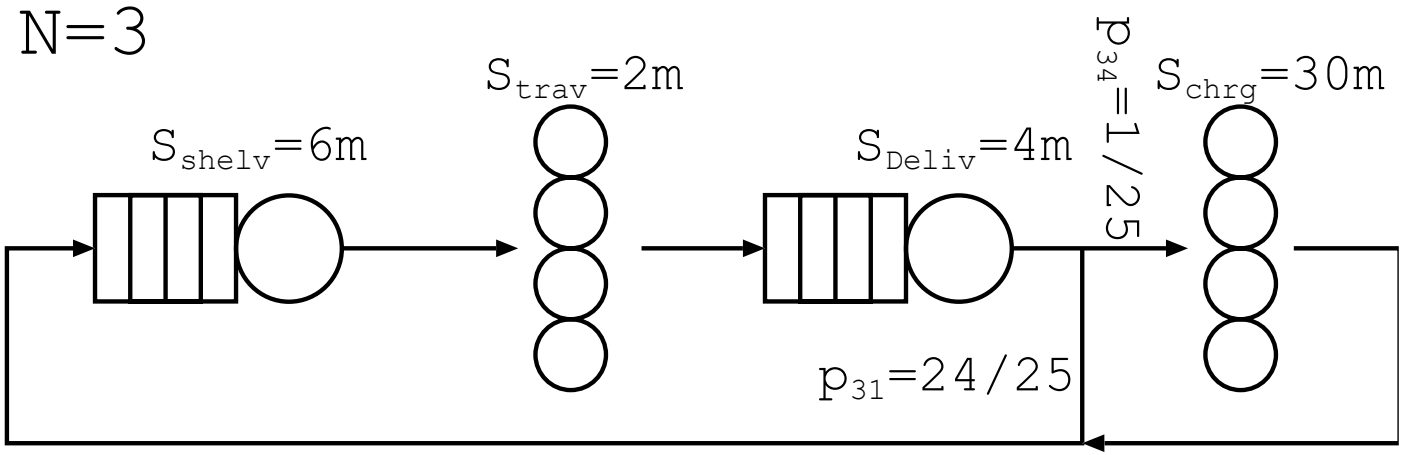
$$R_{Tot} = R_{Sys} + Z \qquad N = X \cdot R_{Tot} \qquad N = X \cdot (R_{Sys} + Z)$$

In particular, Little's law becomes the so-called "*Response Time Law*" which explicitly excludes the think time from the total response time of the system computed in the usual way.

The Response Time Law: $R = \frac{N}{X} - Z$

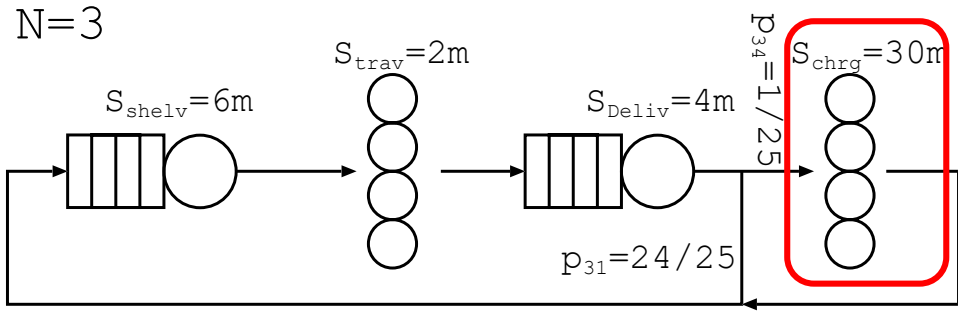
Analysis of Motivating Example

The considered automated warehouse can be modelled with a closed queuing model, with N=3 jobs circulating inside.



Analysis of Motivating Example

To determine the average time between charges, the model is a time-sharing system where station 4 is both the reference and the terminal station, and we compute the system response time with MVA.



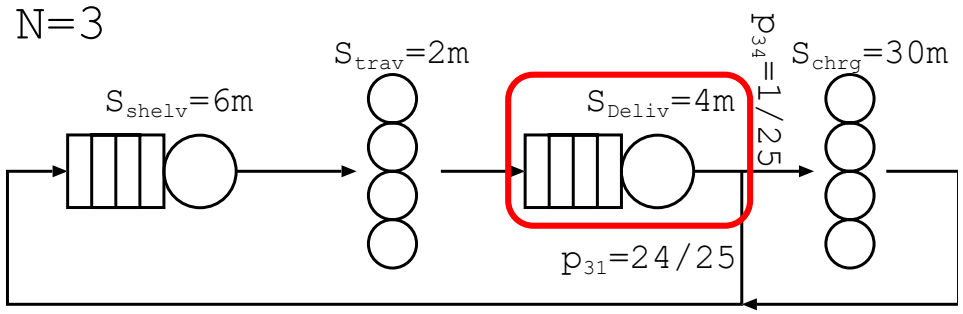
$$\begin{cases} v_1 = 25 \\ v_2 = 25 \\ v_3 = 25 \\ v_4 = 1 \end{cases} \qquad \begin{cases} D_1 = 150m \\ D_2 = 50m \\ D_3 = 100m \\ D_4 = Z = 30m \end{cases}$$

D		150	50	100	Z			30
N	R1	R2	R3	R	X	N1	N2	N3
	0						0	0	0
	1	150	50	100	300	0,0030303	0,45454545	0,15151515	0,3030303
	2	218,181818	50	130,30303	398,484848	0,00466761	1,01838755	0,23338048	0,60820368
	3	302,758133	50	160,820368	513,578501	0,00551898	1,67091671	0,2759491	0,88756473

8,55964168 H

Analysis of Motivating Example

To determine the system throughput, the model is a batch system where station 3 is the reference station. With MVA we determine X.



$$\begin{cases} v_1 = 1 \\ v_2 = 1 \\ v_3 = 1 \\ v_4 = 0.04 \end{cases} \qquad \begin{cases} D_1 = 6m \\ D_2 = 2m \\ D_3 = 4m \\ D_4 = 1.2m \end{cases}$$

D		6	2	4	1,2					
N	R1	R2	R3	R4	R	X	N1	N2	N3	N4
	0							0	0	0	0
	1	6	2	4	1,2	13,2	0,07575758	0,45454545	0,15151515	0,3030303	0,09090909
	2	8,72727273	2	5,21212121	1,2	17,1393939	0,11669024	1,01838755	0,23338048	0,60820368	0,14002829
	3	12,1103253	2	6,43281471	1,2	21,74314	0,13797455	1,67091671	0,2759491	0,88756473	0,16556946

8,27847311 jobs/h

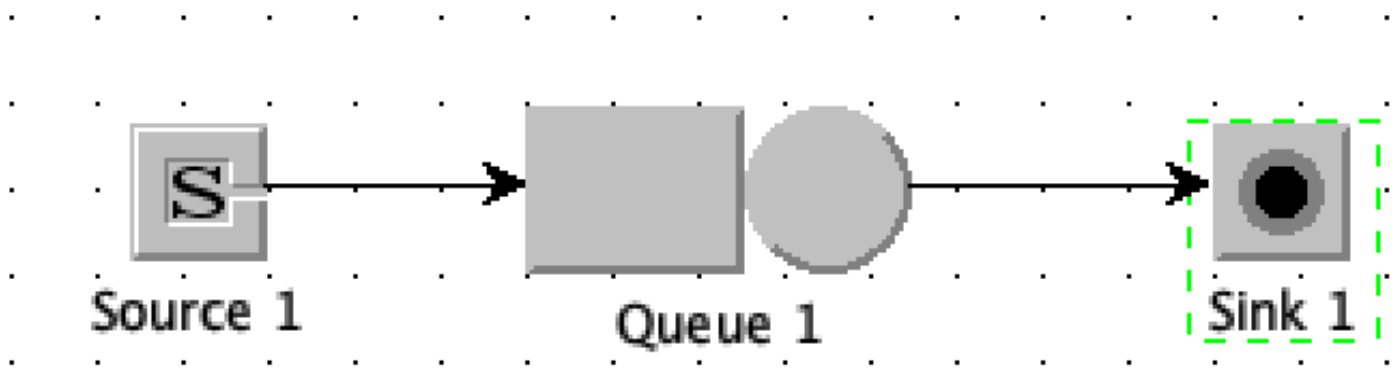
Separable models

Let us consider a simple system composed by a single queue, under different arrival and service processes, and considering different queuing policies.

We consider for both service and arrival process:

- Exponential (c.v. = 1)
- Erlang, with c.v. = 0.5
- Hyper-Exponential, with c.v. = 2

As queuing policies we consider LCFS, PS and FCFS



Separable models

The results obtained using JMT and discrete event simulation for the *Average Response Time (R)* are the following:

Arrival (Average l=0.8 j/s)	Service (Average D = 1s)			
	LCFS			
		Exp	Erl	Hyper
	Exp	4.9845	3.5012	10.9925
	Erlang	3.2733	1.8351	9.2016
	Hyper-exp	10.7959	9.1543	17.2105
	PS			
		Exp	Erl	Hyper
	Exp	4.9813	5.031	4.9152
	Erlang	3.369	2.366	4.1592
	Hyper-exp	10.8972	13.8811	7.8524
	FCFS			
		Exp	Erl	Hyper
	Exp	5.0473	3.5016	11.0676
	Erlang	3.2566	1.8468	9.1904
	Hyper-exp	10.8286	9.0071	17.2209

Average Response Time (R)

Separable models

With the considered arrival rate and demand, analytical results for the M/M/1 model would be:

$$R = \frac{D}{1 - \lambda \cdot D} = \frac{1}{1 - 0.8 \cdot 1} = 5s$$

Note that, beside the M/M/1 case (orange background), such results are also obtained for the PS queuing policy with exponential inter-arrival time, and for M/M/1/LCFS (yellow background).

Arrival (Average l=0.8 j/s)	Service (Average D = 1s)			
	LCFS			
		Exp	Erl	Hyper
	Exp	4.9845	3.5012	10.9925
	Erlang	3.2733	1.8351	9.2016
	Hyper-exp	10.7959	9.1543	17.2105
	PS			
		Exp	Erl	Hyper
	Exp	4.9813	5.031	4.9152
	Erlang	3.369	2.366	4.1592
	Hyper-exp	10.8972	13.8811	7.8524
	FCFS			
		Exp	Erl	Hyper
	Exp	5.0473	3.5016	11.0676
	Erlang	3.2566	1.8468	9.1904
	Hyper-exp	10.8286	9.0071	17.2209

Separable models

Moreover, note that both FCFS and LCFS have basically the same Average Response Time.

Arrival (Average l=0.8 j/s)	Service (Average D = 1s)			
	LCFS			
		Exp	Erl	Hyper
	Exp	4.9845	3.5012	10.9925
	Erlang	3.2733	1.8351	9.2016
	Hyper-exp	10.7959	9.1543	17.2105
	PS			
		Exp	Erl	Hyper
	Exp	4.9813	5.031	4.9152
	Erlang	3.369	2.366	4.1592
	Hyper-exp	10.8972	13.8811	7.8524
	FCFS			
		Exp	Erl	Hyper
	Exp	5.0473	3.5016	11.0676
	Erlang	3.2566	1.8468	9.1904
	Hyper-exp	10.8286	9.0071	17.2209

Average Response Time (R)

Separable models

It has been discovered that the analytical results such the one presented for Open and Closed networks, are valid for a large variety of systems.

In particular, the techniques e can be applied if a system is a *separable queuing network*.

A queueing network is *separable if* it fulfills five main properties.

In the following, we will present the properties directly from the book **Quantitative System Performance** by E. D. Lazoswka.

Assumptions

The first assumption is:

service center flow balance — Service center flow balance is the extension of the flow balance assumption (see Chapter 3) to each individual service center: the number of arrivals at each center is equal to the number of completions there.

Losses clearly violate the *service center flow balance* (1st) assumption, and thus such models are not separable.

Assumptions

The second assumption is:

one step behavior — One step behavior asserts that no two jobs in the system "change state" (i.e., finish processing at some device or arrive to the system) at exactly the same time. Real systems almost certainly display one step behavior.

Assumptions

The third assumption is:

routing homogeneity — To this point we have characterized the behavior of customers in the model simply by their service demands. A more detailed characterization would include the routing patterns of the jobs, that is, the patterns of centers visited. Given this more detailed view, routing homogeneity is satisfied when the proportion of time that a job just completing service at center j proceeds directly to center k is independent of the current queue lengths at any of the centers, for all j and k. (A surprising aspect of separable models is that the routing patterns of jobs are irrelevant to the performance measures of the model. Thus, we will continue to ignore them.)

Assumptions

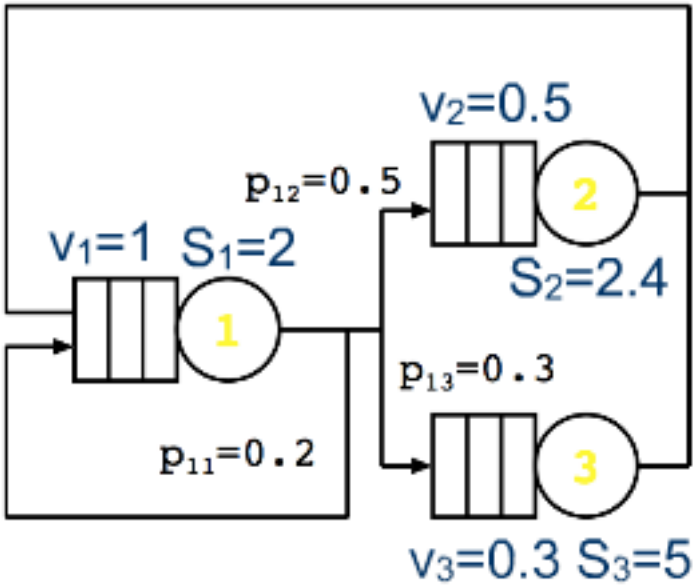
Of the routing policies we have seen, the probabilistic routing fully satisfies the previous assumption.

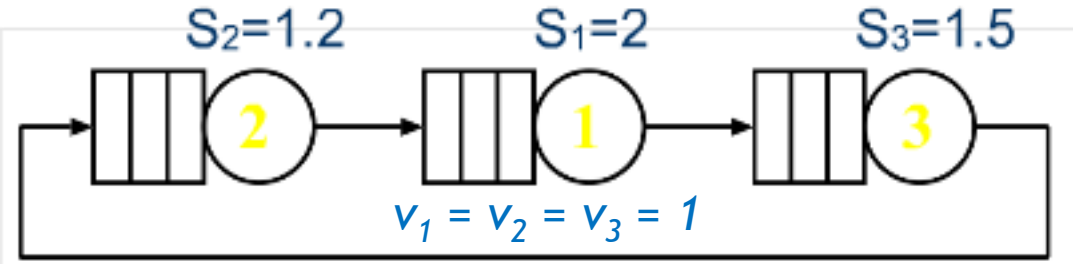
Other routing policies, such as Join the Shortest Queue (JSQ) clearly violate this constraint since the decision of the next station depends on the whole state of the system.

The previous assumption has a very big implication: as long as nodes have the same demands, two queuing networks have the same performance, independently of their topology.

Assumptions

For example, these two networks have exactly the **same performances** in term of *system throughput*, *system response time*, *average number of jobs*, *utilizations* and *residence times*.



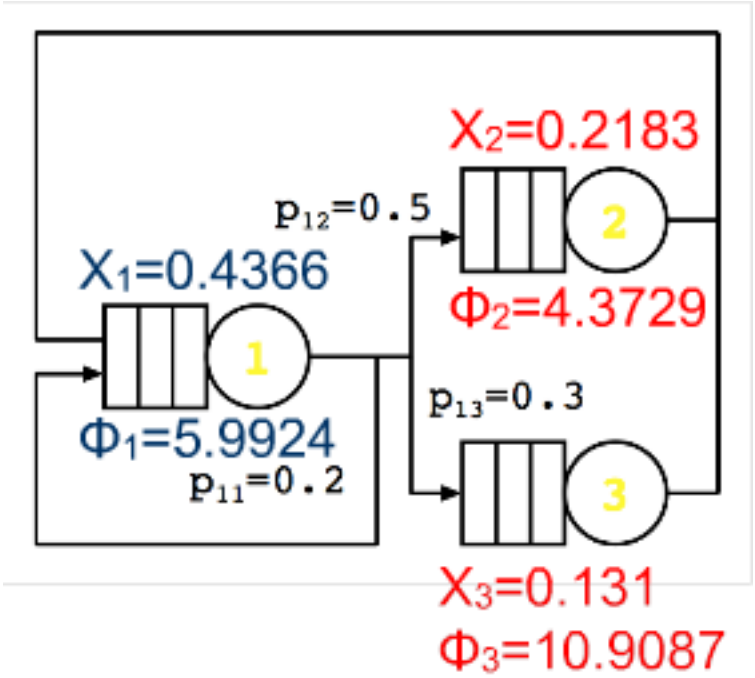


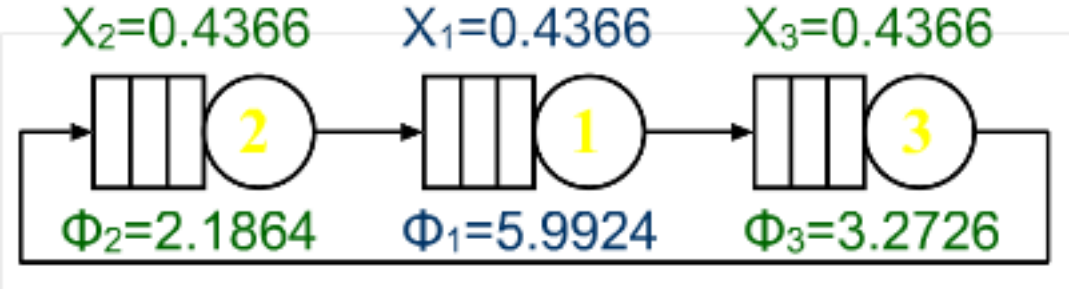
X=0.4366
R=11.4515

U1=0.8732	U2=0.5239	U3=0.6549
N1=2.6165	N2=0.9546	N3=1.4289
R1=5.9925	R2=2.1864	R3=3.2726

Assumptions

However, the two systems have different visits. This means that the two models have **different** *stations throughput* and *resource response times* since these performance indices strongly depend on the visits to the corresponding stations.





Assumptions

In particular, such performance indices are defined as follows:

$$X_k = v_k \cdot X$$

$$\Phi_k = \frac{N_k}{X_k} = \frac{R_k}{v_k}$$

To analyze a queuing network, we need to know its topology just to determine the visits and the demands of the different stations.

We can then perform the analysis using only demands.

Assumptions

The fourth assumption is:

device homogeneity — The rate of completions of jobs from a service center may vary with the number of jobs at that center, but otherwise may not depend on the number or placement of customers within the network.

This is a very tricky and important assumption that considers the combination of service time distributions, type of servers and queuing policies.

Assumptions

The fifth assumption is:

homogeneous external arrivals — The times at which arrivals from outside the network occur may not depend on the number or placement of customers within the network.

Correlated and burst arrivals in open models generally violate this assumption, making them no longer separable.

Arrival and Service time distribution

In general, the distribution alone is not enough to determine whether the properties of separable networks are satisfied or not.

In particular, they might violate or not the *device homogeneity* or the *homogeneous external arrival properties* (4^{th} and 5^{th} respectively).

Non-preemptive queuing policies satisfy the *device homogeneity property* (4^{th}) only for exponential service time distributions.

Finite capacity

Finite capacity almost always do **not** fulfill the *device homogeneity property* (4^{th} assumption).

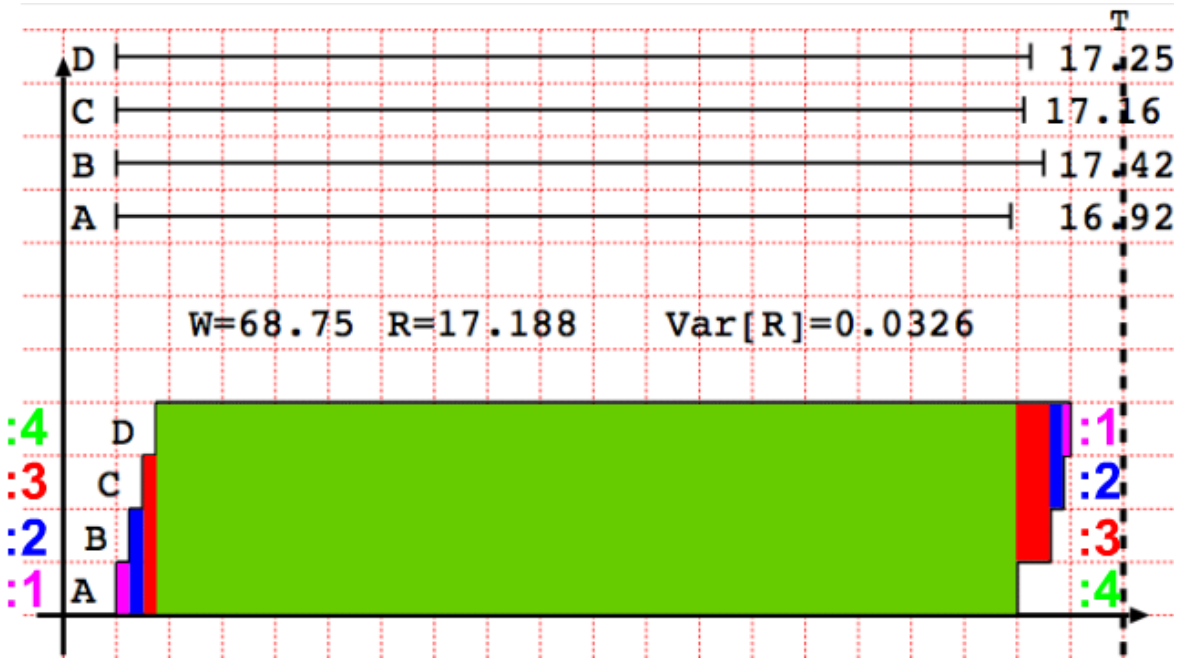
However, for special types of blocking, special network topologies and special service policies, finite capacity models have analytical solution.

Preemptive queuing policies

Preemptive service policies may satisfy the *device homogeneity property* (4^{th}) more often than the non-preemptive ones.

In particular queues with the *Processor Sharing* policy are separable regardless of their service time distribution*.

* Open models have the additional requirement of Poisson arrivals.



Assumptions

If we also add a sixth assumption (that reduces the scope of the fourth) solutions can be computed with simpler algorithms:

service time homogeneity — The rate of completions of jobs from a service center, while it is busy, must be independent of the number of customers at that center, in addition to being independent of the number or placement of customers within the network.

The MVA technique for closed models, and the direct solution for open models, require all 6 assumptions to hold.

Multiple servers and queue length dependencies

Multiple server clearly violates the *service time homogeneity* (6th assumption), so even if separable they require special techniques to be solved.

Infinite server stations (delay centers) instead never create problems in separable models since they do not involve any queuing.

Assumptions: discussion

Even if the previous assumptions seem very strict, they are valid for a large set of models of computer systems.

Moreover, for systems that do not fulfill the previous properties, the considered techniques can provide meaningful approximations.