This is the accepted version of the following article: "Approximating finite resources: An approach based on MVA", which has been published in final form at https://doi.org/10.1016/j.peva.2018.11.005

Approximating Finite Resources: an approach based on MVA[☆]

Vittoria de Nitto Personé and Andrea Di Lonardo

University of Rome Tor Vergata, Via del Politecnico 1 00133, Rome, Italy

Abstract

Blocking is the phenomenon where a service request to a queue is momentarily stopped, but not lost, until the service becomes available again. Blocking is a difficult phenomenon to model analytically, because it creates strong interdependencies between system components. Mean Value Analysis (MVA) is one of the most appealing evaluation methodologies due to its low computational cost and ease of use. In this paper, an approximate MVA for Bloking After Service is presented that greatly outperforms previous results. The new algorithm is obtained by analyzing interdependencies due to the blocking mechanism and by accordingly modifying the MVA equations. The proposed algorithm is tested over about sixty cases and error and computational cost analysis is presented. Finally, the method is applied to capacity planning for healthcare systems, where its simplicity and ease of use is demonstrated, making the case that it can be easily employed by non-experts.

Keywords: blocking, modeling techniques, performance, capacity planning, healthcare systems

1. Introduction

Resources are not infinite. In daily life, the space to host patients or more generally users, is limited. In computer systems and in networks, memory, bandwidth or even virtual machines are finite. Even systems management policies may assume to limit the arrival flow to guarantee QoS requirements, as admission control or concurrency constraints. While finite capacity of resources is a reality, in modeling for prediction or for decision planning the resource capacity is generally assumed as infinite.

In queueing models, finite resources are represented by the $blocking$ mechanism, where a service request is momentarily stopped, but not lost, until the service becomes available again. Consequently, blocking can affect system performance significantly. In the following, we use the terms node, queue or servers as synonyms.

[☆]Declaration of interest: none. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Email address: denitto@ing.uniroma2.it (Vittoria de Nitto Personé and Andrea Di Lonardo)

In general, when a node is full any upstream server is blocked from sending jobs to the full queue. The rules according which this happens correspond to different behaviors and define the so called $blocking\ model$. We point the interested reader to [1, 2, 3, 4] for an extensive bibliography of different blocking mechanisms that model distinct behaviors of blocking in real systems.

Blocking models are used in several fields, spanning from the most traditional as computer systems, communication systems and networks, production systems, to the less traditional ones including software architectures [5], multi-tier applications [6] and also in the emerging area of healthcare systems [7].

Despite its importance, blocking is a difficult phenomenon to model analytically, because it results in strong dependencies in the behavior of system components. As a consequence, the $separability$ property does not hold and an exact solution is complex and computationally expensive. However ignoring the blocking phenomenon can lead to completely erroneous conclusions.

In this paper, we consider blocking within the concept of general topology networks. To the best of our knowledge, there are few results in the literature for general topology networks [1]. Among the analytical techniques for general topologies, Mean Value Analysis [8] still plays an important role. Because of its simplicity and efficiency, MVA is a popular solution technique for applications from disparate fields, including manufacturing [9], military [10], polling systems [11], software engineering [12], system migration [12], web [13, 14, 15, 16], multi-core systems [17], web traffic modelling [18], and theory [19]. Some extensions of the Arrival Theorem [8], the core of MVA, to blocking exist [1]. In particular, an extension of MVA holds for Repetitive Service (RS) blocking [20], as well as for Blocking After Service (BAS) [21, 22]. The interested reader can refer to [22] for other references limited to cyclic networks. Informally, according to RS mechanism, when a node is full an arriving job is rerouted to its sending queue where it receives a new independent service. In this case, queues are never effectively blocked. On the contrary, according to BAS mechanism, a node that completes a job for a full queue is blocked until the destination completes a service.

In [20], a class of product form (PF) networks is considered. PF solutions hold when in the steady state each queue works independently from each other queue, as it was in isolation. According to RS definition, the node is not blocked at all, but the service is repeated until the destination queue becomes "available" again. This allows for the separability property.

In [21, 22], non-PF networks with BAS blocking are considered. In [22], the proposed extension, called EMVA, is obtained using load dependent servers and queue length distributions. The standard MVA technique suffers of instability which is more pronounced as population grows. In [21], the MVABLO algorithm tries to modify the arrival theorem to include the blocking behavior without resorting to the queue length distribution. In some cases, the results show that throughput is effectively modeled, but other performance measures are not captured well. MVABLO shows the worst results for cyclic networks, that is queues connected in a unidirectional ring topology.

On the basis of the above considerations, we present an approximate MVA for BAS blocking that iteratively corrects node populations to consider the effects of blocking. The method is called debeMVA (MVA for dependent behaviour) and is obtained by an in-depth analysis of the interdependencies due to the blocking mechanism and by

consequently modifying the MVA equations. The crux of the proposed approximation is a) the identification of nodes mainly influenced by blocking and b) a modification of their population that is computed iteratively. Note that blocking leads to a not negligible transient behaviour immediately before/immediately after saturation. We define several functions that capture the dependency between pair of nodes and we propose an iterative approach to capture transient behaviour in the long term. We propose a conjecture to approximate the Arrival Theorem for dependent behaviour. debeMVA shows good results for general topology networks and improves significantly the approximation regarding of MVABLO [21]. In all cases, convergence speed is fast.

As a case study we select a real problem where the ease of use and low computational complexity are demonstrated. We consider healthcare systems that represent a critical and important sector of social life. In this case, capacity planning studies are very important both for care efficiency and economical perspective. Moreover, often decisions must be taken by non-expert people and the ease of use and intuitive appeal of MVA is crucial for its usability. We apply the proposed method to the capacity planning for healthcare systems presented in [7]. We obtain the same results, but at a lower computational cost and with a more flexible approach. We show that even a non-expert can easily investigate all options and select the most convenient and efficient solution.

The paper is organized as follows. Section 2 presents the model and the characteristics of blocking. Section 3 presents related works and introduces the core characteristics of debeMVA. Section 4 presents the proposed approach, the Blocking Arrival Theorem, debeMVA algorithm and the results of the approximation on three networks with high correlation among nodes, so emphasizing the difficulty of approximating a non-separability behaviour. Section 5 presents computational cost and error analysis of experiments on 56 networks that differ in the number of nodes, their topologies, and their parameters. We compare results obtained by the proposed algorithm with results obtained by MVABAS [23], MVABLO [21] and the exact solution. Finally, Section 6 showcases the method using a case study. Section 7 concludes the paper.

2. The Model

We consider a single-class closed queueing network with routing matrix $\boldsymbol{P}$ such that jobs departing from queue i are directed to queue j with probability p_{ij}. The network include abstract scheduling single-server nodes and IS nodes. The service time s_i is assumed exponentially distributed for single-server nodes and generally distributed for IS nodes. By denoting the capacity of queue j and its current population with B_j and n_j respectively, when $n_j = B_j$, then queue j is *full* and does not accept in its waiting buffer any *new* job before a departure occurs. In this case, queue i essentially becomes *blocked*. Note that deadlock can occur in the presence of blocking. In the following, we assume that the network population value is such that deadlock prevention is guaranteed [1]. The time of blocking, the unblocking rule and the behavior of the job possibly in execution in i at the blocking time are defined by the blocking mechanism.

Blocking After Service (BAS) is one of the most used blocking mechanisms and is defined as follows: a queue i, if not empty, processes a job regardless of the job population at its destination j. When node i completes service and node j is full, node i suspends any activity (i.e., it is blocked) and the completed job waits until a departure

occurs from node j. At that moment two simultaneous transitions take place: the completed/blocked job moves from i to j (since j can now accept a job, i "unblocks") and the job that leaves j (which effectively "unblocks" server i). In a general network topology where more than one queue compete for sending a job towards a full queue j, a policy regulating the order in which queues unblock has to be defined. Usually, the First Blocked First Unblocked (FBFU) policy is considered fair: first unblock the queue that was blocked first. BAS is used to model production systems and disk I/O subsystems [24].

The blocking mechanism and the simultaneous transitions yields a complexity in the state space transitions as can be seen by the simple example of a three nodes network, see Fig.1. Node 1 has finite capacity with $B_1 = 1$. Its state space is shown in Fig.2, where we denote with μ_i the node i service rate. In this case, the simultaneous transitions can occur only between pairs of nodes (1,2 or 1,3). For example, state (1 [3,2] 1 1) denotes the case where 1 job is in each node, node 1 is full and both nodes 3 and 2 have been blocked according to this order. As soon as node 1 completes the job, e.g. for node 3, two simultaneous transitions take place from 1 to 3 and from 3 to 1. As a consequence, the arrival state is (1 [2] 1 1), where just node 2 is still blocked.

Although the state space is small, it lacks the necessary lattice structure on its boundary part that is a condition for Product Form solutions [25]. Note that in a small case the boundary part is the most of the whole state space and this means that small cases can be more critical than big ones in capturing nodes behaviour.

In Section 4, we derive an approximate iterative MVA-based method that effectively captures the complexity of the behaviour of the queues due to their dependencies and simultaneous transitions.

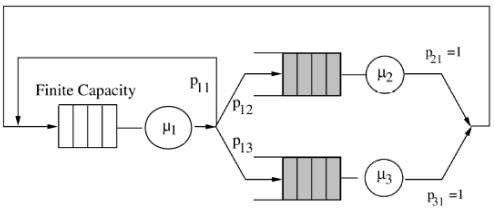


Figure 1: A small queueing network

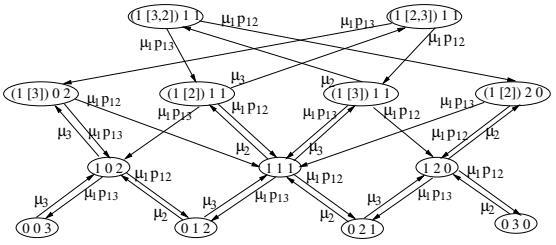


Figure 2: The state space transitions

3. Related work

As introduces in Section 1, [21] presents a solution for BAS blocking based on MVA. The algorithm is called MVABLO and tries to modify the arrival theorem to include the blocking behavior. In some cases, the results show that throughput is effectively modeled, but other performance measures are not captured well. As soon as the congestion grows the results exacerbate, especially for residence times and mean queue lengths. The results fluctuate between over- and under-estimating the performance metrics, so making difficult error prediction. Cyclic networks further exacerbate this effect because of the nature of the flow.

A possibility for the error source is identified by the author himself and resides in the "method" to recognize a saturation condition. Indeed, using the *mean* queue lengths is too late to capture the blocking condition: a blocking condition can appear even when the mean queue length is less than the node capacity. In fact, it is enough that the total network population *N* is greater than B_j to produce blocking conditions in sending nodes of node j. To overcome this problem, our method catches the saturation before the mean population exceeds the node capacity.

However, this is not the main error cause, but we envisage the following most important one: the "effective intensity" of the blocking is not captured and the correction is operated always in the same measure by simply canceling the arriving job to the full node. On the contrary, it is worth noting that the blocked job cannot disappear from the network, but it has to wait in the sending node, accordingly to BAS blocking. To this aim, we introduce the *Arrival Job* (AJ) function. This function captures the dependence between different nodes and is iteratively computed. So our method is able to model the non-separability characteristic. On the other hand, blocking frequency depends on workload. Indeed, as *N* grows, blocking will occur more frequently. In the following, we introduce the *blocking intensity* function to capture the effective incidence of the blocking phenomenon at each iteration step. As the results show, this offers a correction on the limitation of MVA that deals exclusively with mean values.

Moreover, in [21] just nodes directly involved in blocking are considered. As we show in next section, blocking affects also different nodes not directly involved in blocking and the AJ function is evaluated for all such nodes.

We defer to the end of next section for a complete comparison of our method with MVABLO.

4. MVA for dependent behaviour

The Mean Value Analysis algorithm [8] first derives the node mean response time and then computes node throughput and mean population by using Little's equation. The mean response time can be iteratively derived based on the Arrival Theorem [26], [27].

In this paper, we propose an approximate MVA by modifying the two fundamental steps of the algorithm: the Arrival Theorem and the related response time equation. Due to blocking we conjecture that the mean population at arrival time is not identical to the mean population at any time as for the separable network. Specifically, the mean

population at arrival time is iteratively computed and depends on the role of the node within the blocking phenomenon.

As a first step, we define the role a node can play with respect to the blocking occurrence. This is the key to capture the dependent behaviour among nodes. This is presented in Section 4.1.

As a second step, we present the proposed new Arrival Theorem, see Section 4.2. We show that the response time equation in MVA needs to be modified as follows: 1) for nodes affected by the blocking mechanism, the arrival job can not be taken into account as a constant 1, but its value needs to be iteratively computed (Section 4.2); 2) the response time must include the blocking time when the node may be blocked. We present in Section 4.3 this second important step of the algorithm.

In the following subsections, we show how the proposed approximation works on three networks [23] characterized by high correlation among nodes, so emphasizing the difficulty of approximating a non-separability behaviour. As we observed in Section 2, it is worth noting that small networks represent the most difficult cases. Indeed, interdependences among nodes is stronger than in large networks. This is also evident from the boundary part of the state space that is the most of whole state space (see Fig.2). In [28] we show detailed experimentation that confirm the results obtained for these three cases.

4.1. The non-separability condition

The arrival theorem states that the average node population at arrival instants is the same as the average node population at any time, considering the network with one job less. This is true for a separable network. When the separability condition does not hold, a complex dependence affects the behaviour of connected nodes: the average node population at arrival instants cannot grow over the node capacity and, on the other hand, the average population of sending nodes of a full queue can grow more than in the corresponding infinite case. Moreover, this dependence can propagate to more than two nodes and even to nodes not directly involved in the blocking. We propose a Blocking Arrival Theorem that includes an Arrival Job (AJ) function for taking into account this complex dependency among the nodes. AJ function determine the value of the arrival job, taking into account the blocking intensity and the dependence between two connected nodes. This function is iteratively computed.

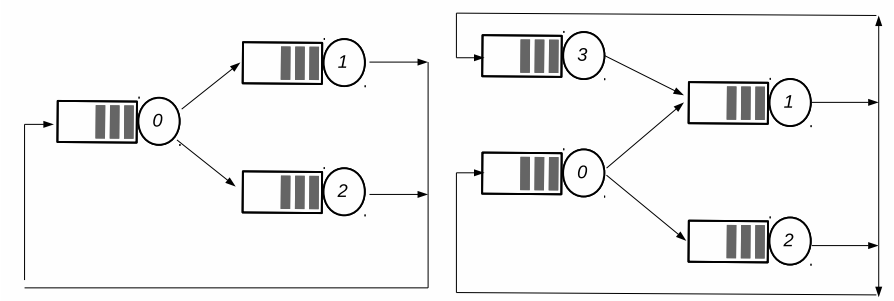
First we define when a node can be involved in a blocking occurrence:

Definition 1. *Blocking Involved Node (BIN node)* A node is involved in blocking if it is in at least one of the following conditions:

- f_BIN: the node has finite capacity;
- s_BIN: the node can be blocked, i.e., the node is a sending node to a finite capacity node;
- d_BIN: the node is a destination of a node that can be blocked.

With "involved" we mean that node performance are influenced by a blocking occurrence. While the first two definition cases are quite obvious, the third case is perhaps

less intuitive. Let us consider a closed queueing network and focus on three nodes involved in the blocking occurrence: the finite capacity node f_BIN, its sending node s_BIN, and a destination node d_BIN of s_BIN different from f_BIN. If all node capacities were infinite (IC), in the long term the node mean population would be uncorrelated according to the separability characteristic. In case of finite capacity, the population flows in these three nodes are correlated: during the time f_BIN is full its population is fixed while the s_BIN population can grow more than in the IC case. In the long term, this also affects the mean population in d_BIN. Note that the individuation of d_BIN node is crucial for the approximation, indeed it captures the propagation of blocking effects in the network. In next section, we show this particular behaviour of d_BIN nodes for the following two networks with parameters chosen to emphasize this effect.



(a) Network 46 (b) Network 0

Figure 3: Two example networks

Example 1

In Fig.3 we show two topologies that represent especially difficult cases since all nodes can be BIN nodes. In Fig.3a, network 46 is a central server topology and if only node 2 has finite capacity it is a f_BIN, node 0 is a s_BIN and node 1 is a d_BIN. If all nodes have finite capacity, then they all are f_BIN and according to network population each node can play several roles at the same time. We exacerbate this condition by adding a further sending node to node 1, see Network 0 in Fig.3b. This case is particularly useful to show the robustness of the proposed solution, as we show in next section.

Example 2

Let us consider the two networks in Fig.3. Network 46 is a central server topology with parameters B_j and $E[s_j]$ shown in Table 1. It is easy to be convinced that this is a highly unbalanced case: node 2 is both the bottleneck and the smallest capacity node while its sending node 0 is the fastest node. This is confirmed by the blocking probabilities computed by exact solution and shown in Fig.4a: node 0 shows a sudden grow of this probability even for small population values while the other nodes are never blocked. In the last column of Table 1, we show the blocking probabilities for the maximum admissible population ($N = 39$) for deadlock free operation [1]. We

Table 1: Network 46: a central server with 3 queues

node	B_j	$E[s_j]$	pb_j
0	35	0.1	0.966667
1	20	2.5	0.000000
2	5	6.0	0.000003

Table 2: Network 18: a cyclic network with 5 queues

node	B_j	$E[s_j]$	pb_j
0	4	20.1	0.073476
1	3	2.0	0.907808
2	5	10.0	0.539043
3	4	2.1	0.903199
4	2	5	0.769521

observe that the networks with such unbalanced conditions are the most difficult to approximate. To further stress the proposed approximation, we modify the topology by adding node 3 as a sending of node 1 (see Fig.3b) with the same parameters $B_3 = B_1 = 20$ and service time $E[s_3] = E[s_1] = 2.5$, so to have a balanced behaviour between these two nodes. Fig.4b shows the blocking probabilities where there is still a big difference among the blocking behaviour of node 0 and the other nodes blocking, but there is one other node (queue 3) with not zero blocking probability.

Example 3

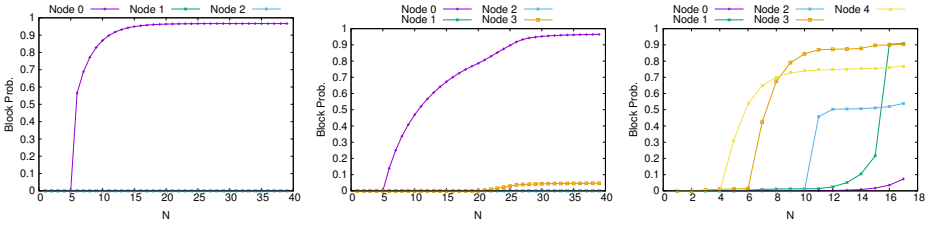
The last example is network 18, a five nodes cyclic network, which is quite unbalanced as can be seen by the node parameters B_j and $E[s_j]$ shown in Table 2. In order to avoid deadlock, the maximum population is $N = 17$. For this value, each queue reaches the maximum blocking probability that is shown in the last column of Table 2, evaluated by exact solution. In Fig. 4c, the blocking probabilities confirm that this is a particularly hard case: all nodes experience blocking and the simultaneous transitions may involve all nodes.

Blocking occurrence is not a "static" condition derived just by node characteristics (e.g. routing and capacity), but it is also determined by the workload, i.e. the network population k. As a consequence, we define a *blocking intensity* function $\beta_j(k)$ as follows:

Definition 2. *Blocking Intensity function (BI function)*

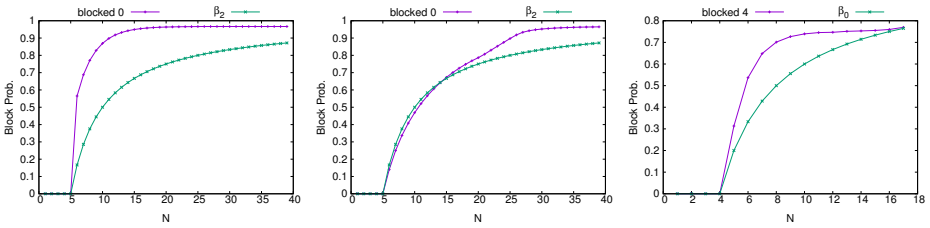
$$\beta_j(k) = \begin{cases} 1 - \dfrac{B_j}{k} & \text{if } k > B_j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The BI function of a node j is a good approximation of the node blocking probability of its sending nodes. Indeed, the ratio between node capacity and network population determines how frequently sending nodes to j will be blocked. For example, if B_j approaches k sending nodes to j will be never blocked ($\beta_j(k)$ approaches 0), while if $B_j << k$ sending nodes will be very often blocked ($\beta_j(k)$ approaches 1). In general, blocking probability can be computed just analytically with high computational cost and complexity. BI function requires low-cost computation.



(a) Network 46 (b) Network 0 (c) Network 18

Figure 4: Blocking probabilities



(a) Node 0 blocking in Net 46 (b) Node 0 blocking in Net 0 (c) Node 4 blocking in Net 18

Figure 5: Blocking probabilities and BI function

Finally, we show how the BI function $\beta_j(.)$ can predict quite well the blocking probability of a sending node. Fig. 5 compares $\beta_j(.)$ with the blocking probability of its sending node first blocked. Specifically Figs. 5a and 5b show the blocking probabilities for node 0 and $\beta_2(.)$ for network 46 and 0 respectively. Note that network 46 is more difficult than network 0, since blocking suddenly surges when the population grows from N=5 (no-blocking) to N=6 with a blocking probability for node 0 of about 0.56 (see Fig. 4b). Nevertheless, for high values of population the estimation is still good even in this challenging case: for $N = 35$, $\beta_2(k)$ is about 0.86 while the blocking probability on its sending node is 0.96. Fig. 5c shows similar behaviour for Network 18 and its first blocked node, i.e. node 4 . For $N = 17$ the estimation is quite precise, while for the lowest values of population the error is more pronounced.

The results for 56 networks in [28] confirm this characteristic of BI functions.

4.2. Arrival Theorem for dependent behaviour

The crux of the proposed approximation is the computation of the mean value of an arrival job. To this end, we define a node function to evaluate the mean value of the arrival job to a BIN node. The function is iteratively computed as soon as a *capacity violation* is detected, i.e. the mean population exceeds the node capacity, and until the violation is removed:

Definition 3. *Arrival Job function (AJ function)* The AJ function is initialized to $z_h(0) = 1$, for each node h. In each iteration step k where a f_BIN node j violates its capacity, the AJ function $z_j(k)$ is updated as follows:

$$z_j(k) \leftarrow z_j(k) - \beta_j(k) \cdot z_j(k) \tag{2}$$

and correspondingly, for each s_BIN node i of j, $z_i(k)$ is:

$$z_i(k) \leftarrow z_i(k) + \beta_j(k) \cdot p_{ij} \cdot z_j(k) \tag{3}$$

where k is the network population.

Note that for the solution consistency, the AJ function $z_j(k)$ is initialized with 1 for *all* nodes, including the non-blocking ones, and then iteratively computed when a capacity violation is detected. The update is repeated until the violation is removed. It is worth noting that equation (3) captures the behaviour dependence between two connected nodes i and j, which is expressed by AJ functions $z_i(k)$ and $z_j(k)$. This represents the approximation of the non-separability condition of the blocking network. Using AJ function, we conjecture the following *Blocking Arrival Theorem*:

Conjecture 1. For a closed queueing networks with BAS blocking and k jobs, the mean population $A_j(k)$ for a node j at arrival instants can be defined as follows:

$$A_j(k) = E[n_j(k-1)] + z_j(k) - 1 \tag{4}$$

In case of blocking, the AJ function $z_j(k)$ corrects opportunely the mean population at arrival instants. Its initial value (i.e. 1) does not change for nodes not involved in blocking. As a consequence, in this case equation (4) coincides with the exact Arrival Theorem. Note that when $z_i(k) > 1$ the mean population $A_i(k)$ is greater than the standard case ($z_i(k) = 1$). This is consistent with the blocked condition (second condition in Definition 1) and therefore the queue can grow. Moreover, the increment value in (3) is a function of the full queue j and of the percentage of times node i sends jobs towards the full queue (p_{ij}). On the other hand, when $z_j(k) < 1$ the mean population $A_j(k)$ is less than the standard case ($z_j(k) = 1$) and this is consistent with the finite capacity.

As a final remark, it is important to reflect on the fact that MVA determines mean values. The blocking condition can arise in a network much earlier than when the average node population approaches its node capacity. The ability of identifying the most relevant conditions for blocking occurrence is the crucial step of the approximation. This was one of the major problem of the previous methods [21], [23] that affected

negatively the approximation quality. We overcome this problem by computing the AJ functions (2) and (3) as soon as the network population reaches a critical threshold value. The extended experimentation [28] showed that node velocity is more important than node capacity in determining a blocking occurrence. The following example illustrates the threshold definition.

Example 4

Consider again network 18 in Example 3. While one could think that node 3 was the first blocked node, since it is sending to the minimum capacity node 4 ($B_4 = 2$), from the exact results of the blocking probabilities (see Fig.4c) one can easily see that node 4 is the first to experiment blocking since it is sending to the slowest node 0.

Extended experimentation in [28] confirms the suggestion of Example 4 and our algorithm starts as soon as the network population reaches the threshold value $B_j + 1$ where j is the finite capacity node with the maximum service time, that is the slowest node. Eq. (2) is applied to node j and eq. (3) to all its sending nodes. During algorithm iterations, as soon as a mean population exceeds node capacity the AJ functions are opportunely applied until the capacity violation is removed. The number of capacity violations θ is computed during iterations. It measures the extra computation required with respect to traditional MVA, see Sect.5. In conclusion, the function $z_j(k)$ is computed for each node j as follows:

1. for each node h, the starting value is $z_h(0) = 1$
2. initialization phase: as soon as network population reaches the threshold value $B_j + 1$, where j is the finite capacity node (f_BIN) with the maximum service time (i.e. the slowest node) $z_j(k)$ is updated by equation (2) and all its sending nodes (s_BIN) by equation (3); for all destination nodes l (d_BIN) of node i such that $l \neq j$, then i and l are updated with eq. (2) and eq. (3) respectively;
3. iterative phase: as soon as a capacity violation is detected for a node j, that is the mean queue length $E[n_j(k)]$ violates the queue capacity B_j, both equations (2) and (3) are applied to node j (f_BIN) and all its sending i (s_BIN) respectively, until the violation is removed.

The complete algorithm is presented in Section 4.4. The improvement obtained by the initialization phase is shown in the next section where we present the second important step of the proposed Approximate MVA.

4.3. Blocking time and mean response time

The MVA algorithm starts from the mean response time computation and iteratively computes throughput and mean queue length. However in a network with blocking, the node mean response time can be longer than in the corresponding infinite capacity network for two reasons:

- according to the BAS mechanism, the node i activity is stopped when it sends a job to a full queue j; the activity will be resumed as soon as the full queue completes a service and other nodes eventually blocked before i have resumed their activities (FBFU policy);

- during the node blocking time, its queue length can grow, thus increasing the mean residence time.

The first term is considered by defining a blocking time $BT_i(k)$ for each node i with a full destination node, that is then added to the node response time. The second term is simply derived by using longer response times in the Little's equation for the mean population of the blocked nodes. The blocking time $BT_i(k)$ is iteratively computed as defined in the following:

Definition 4. *Node Blocking Time (BT)* For each node h, its blocking time is initially set to $BT_h(0) = 0$. In iteration step k where a f_BIN node j violates its capacity, for each of its s_BIN node i the blocking time is updated as follows:

$$BT_i(k) \leftarrow BT_i(k) + p_{ij} \cdot \nu_{ij} \cdot E[s_j] \tag{5}$$

where k is the network population and ν_{ij} is a *relative velocity factor* defined as follows:

$$\nu_{ij} = \frac{E[s_j]}{E[s_j] + E[s_i]} \tag{6}$$

Note that for the solution consistency, the node BT is initialized with 0 for all nodes, even not BIN node, and then iteratively computed until the capacity violation is removed. Next section presents the complete algorithm.

For exponential service time, the node BT is a function of the service time of the full node [21]. In addition, the service time of the full node j is weighted with the workload percentage that from node i proceeds towards node j, that is the routing probability p_{ij}. Moreover, it is easy to be convinced that the waiting in node i blocked by j, depends on the relative node "velocity": a node h with a big service time in respect of its destination j, it is probably less blocked than a sending node i with shorter service time. It is also possible than during the time j is full, "slow" node h does not complete a job. This is also captured by the relative velocity factor. Note that when node j is much slower than its sending node i ($E[s_j] \gg E[s_i]$) the factor ν_{ij} approaches 1, thus allowing the increase of the node i blocking time by the node j service time. On the other hand, when node j is much faster than its sending node i ($E[s_j] \ll E[s_i]$) the factor ν_{ij} approaches 0, thus canceling the contribution of the node j service time.

In conclusion, the node BT is used for the computation of the mean response time $E[t_j(k)]$ as follows [23]:

$$E[t_j(k)] = \begin{cases} E[s_j] + \dfrac{BT_j(k)}{k} & \text{delay center} \\ E[s_j](z_j(k) + E[n_j(k-1)]) + BT_j(k) & \text{single server center} \end{cases} \tag{7}$$

Finally, it is worth noting that the blocking time (5) is again a function representing a dependence between two nodes. Although the dependence does not allow for separability, we can capture its effect by using the definitions in equations (3), (5) and (6)

Table 3: Symbols Table for node j and network population k

$A_j(k)$	mean population at arrival instants
B_j	node capacity
$\beta_j(k)$	node Blocking Intensity function
$BT_j(k)$	node Blocking Time
$n_j(k)$	node population
p_{ij}	routing probability
ν_{ij}	relative velocity factor of blocked node i vs full node j
s_j	node service time
$t_j(k)$	mean node residence time
$X_j(k)$	mean node throughput
$z_j(k)$	Arrival Job function

in the MVA algorithm, as experimentation shows. Note that Equation (5) is different from the one proposed in [21] and in [23]. We summarize all differences comparing to previous methods in the next section.

4.4. debeMVA

Algorithm 1 and Algorithm 2 present the debeMVA algorithm. It is worth noting the last step of Algorithm 1. Since the iterative characteristic of the method, the BIN node population can be updated several times as the node plays different roles. As a consequence, mean population of node j at step k may be less than previous value at step $k-1$. In this case, we force $E[n_j(k)] \leftarrow E[n_j(k-1)]$ by reducing its upstream nodes population proportionally with routing probabilities.

We apply debeMVA to networks 46, 0 and 18 in Examples 1 and 3, and compare the results with those obtained by MVABAS[23], MVABLO [21] and the exact solution. In the following, we present the performance results while we defer the error and cost analysis to Section 5. As already discussed in Example 2, Network 46 is quite unbalanced, with node 2 that is both the bottleneck and the smallest capacity node, while its sending node 0 is the fastest node. We show that the proposed method improves the results over the previous algorithms and captures quite well the behaviour of all nodes, see Fig.6.

To further stress the proposed algorithm, we consider Network 0 in Fig.3b. In this case, the situation is even more complicated since node 1 can be full and several iterations could fail, by moving "blocked jobs" from node 2 to node 0, from node 0 to node 1, but also from node 1 to node 3. Nevertheless, as Fig.7 shows, our algorithm captures population of nodes 0, 1 and 2 quite well and much better than previous algorithms. It is worth noting that population and residence time of node 3 are not well estimated by debeMVA, neither by past methods. This confirms our intuition regarding this case as stressful for the proposed solution and open to further investigations.

As discussed in Section 4.2, the effects of blocking are not limited to the pairs (s_BIN, f_BIN) nodes. In other words, the non-separability condition extends to destinations of blocked nodes (d_BIN). This is particularly evident on central server topologies where an increase of mean population is observed on destinations of blocked

ALGORITHM 1: debeMVA

Input: P (routing matrix), $E[\overline{s}]$, $\overline{B}$, N (jobs in the network), M (number of stations)
Output: $E[\overline{n}(N)]$, $E[\overline{t}(N)]$, $E[\overline{X}(N)]$, θ (capacity violation)
$\overline{z}(1) \leftarrow \mathbf{1}$, $\overline{BT}(1) \leftarrow \mathbf{0}$, $repeat \leftarrow 0$ $\overline{a} \leftarrow$ **false**, $\theta \leftarrow \mathbf{0}$;
for $k \leftarrow 1$ **to** N **do**
  Algorithm2(P, $E[\overline{s}]$, $\overline{B}$, k, $\overline{a}$)
  repeat
    $repeat \leftarrow 0$
    `// calculate` $E[\overline{t}(k)]$
    for $j \leftarrow 1$ **to** M **do**
      if *j is delay center (IS)* **then**
        $E[t_j(k)] \leftarrow E[s_j] + \dfrac{BT_j(k)}{k}$
      else
        $E[t_j(k)] \leftarrow E[s_j] \cdot (z_j(k) + E[n_j(k-1)]) + BT_j(k)$
      end
    end
    `// calculate` $E[\overline{X}(k)]$
    for $j \leftarrow 1$ **to** M **do**
      $E[X_j(k)] \leftarrow \dfrac{k}{\sum_{i=1}^{M} E[t_i(k)] \cdot e_i/e_j}$
    end
    `// calculate` $E[\overline{n}(k)]$
    for $j \leftarrow 1$ **to** M **do**
      $E[n_j(k)] \leftarrow E[t_j(k)] \cdot E[X_j(k)]$
      `// capacity violation`
      if $E[n_j(k)] > B_j$ **then**
        $repeat \leftarrow 1$
        $\theta \leftarrow \theta + 1$
        foreach *upstream node i IS or not full* **do**
          $z_i(k) \leftarrow z_i(k) + p_{ij} \cdot z_j(k) \cdot \beta_j$
          $BT_i(k) \leftarrow BT_i(k) + p_{ij} \cdot E[s_j] \cdot \nu_{ij}$
        end
        $z_j(k) \leftarrow z_j(k) - z_j(k) \cdot \beta_j$
      else
        `// if no capacity violations are detected`
        if $E[n_j(k)] < E[n_j(k-1)]$ **then**
          $E[n_j(k)] \leftarrow E[n_j(k-1)]$
          foreach *upstream node i* **do**
            $E[n_i(k)] \leftarrow E[n_i(k)] - \dfrac{p_{ij} \cdot (E[n_j(k-1)] - E[n_j(k)])}{\sum_{l=1}^{M} p_{lj}}$
          end
        end
      end
    end
  until *repeat=1*;
end

ALGORITHM 2: debeMVA: Initialization phase function

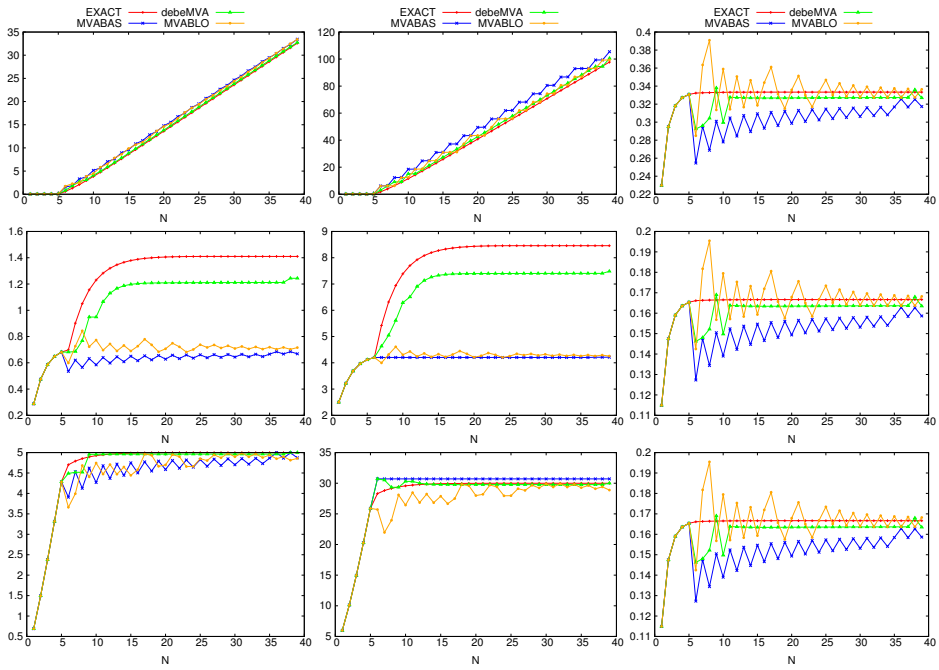
Input: P (routing matrix), $E[\overline{s}]$, $\overline{B}$, k (jobs in the network), $\overline{a}$ (true if first correction on upstream node is already done)
Output: $\overline{BT}(n)$, $\overline{z}(n)$, $\overline{a}$

```
foreach station j with the maximum E[s_j] do
    // correct only the first time
    if k > B_j + 1 and a_j = false then
        a_j ← true
        foreach upstream node i of node j do
            z_i(k) ← z_i(k) + p_ij · z_j(k) · β_j(k)
            BT_i(k) ← BT_i(k) + p_ij · E[s_j] · ν_ij
        end
        z_j(k) ← z_j(k) − z_j(k) · β_j(k);
    end
end
foreach station j with the maximum E[s_j] do
    if k > B_j + 1 then
        foreach upstream node i of node j do
            nd ← number of downstream node of i
            // no cyclic network
            if nd > 1 then
                foreach downstream node l of node i (l ≠ j) do
                    z_l(k) ← z_l(k) + z_i(k) · β_j(k) · p_il
                end
                z_i(k) ← z_i(k) − z_i(k) · β_j(k)
            end
        end
    end
end
```

nodes. For both networks 46 and 0, as soon as the population grows to 6, the population of node 1 suddenly grows until it is full (see second rows in Fig.6 and 7), even if the node is practically never blocked (see Fig.4a and 4b). Note that previous methods are not able at all to predict mean population and residence time of node 1 for both networks. Our algorithm captures well this challenging behaviour.

Finally, we consider Network 18. As described in Example 3, this is a particularly hard case since all nodes experience blocking and simultaneous transitions may involve all nodes. Even in this case, our algorithm is able to catch the behaviour of each node. In Fig.8-12 we show the results for all nodes. Even for this particularly difficult case, our algorithm dramatically improves on the approximate performance metrics comparing to past methods.

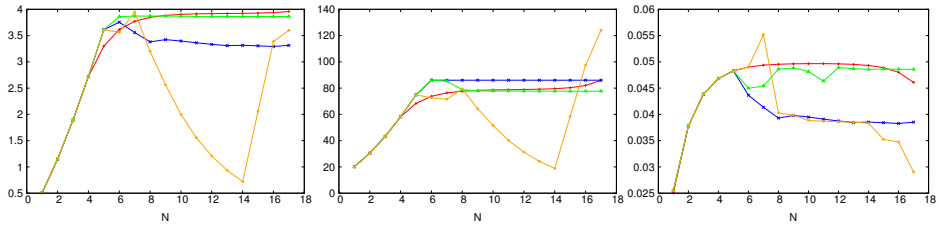
All three cases above show that the method shares some uncertainty in prediction as soon as blocking occurs. This is intrinsic to an approximate iterative method. Indeed, when the population exceeds a given threshold, the results stabilize and the approximation is good. Specifically for throughput prediction, this happens in Fig. 6 for network 46 when $N > 10$ and in Fig. 7 for network 0 when $N > 17$. Network 18 is a quite



(a) Mean Population (b) Mean Residence (c) Throughput

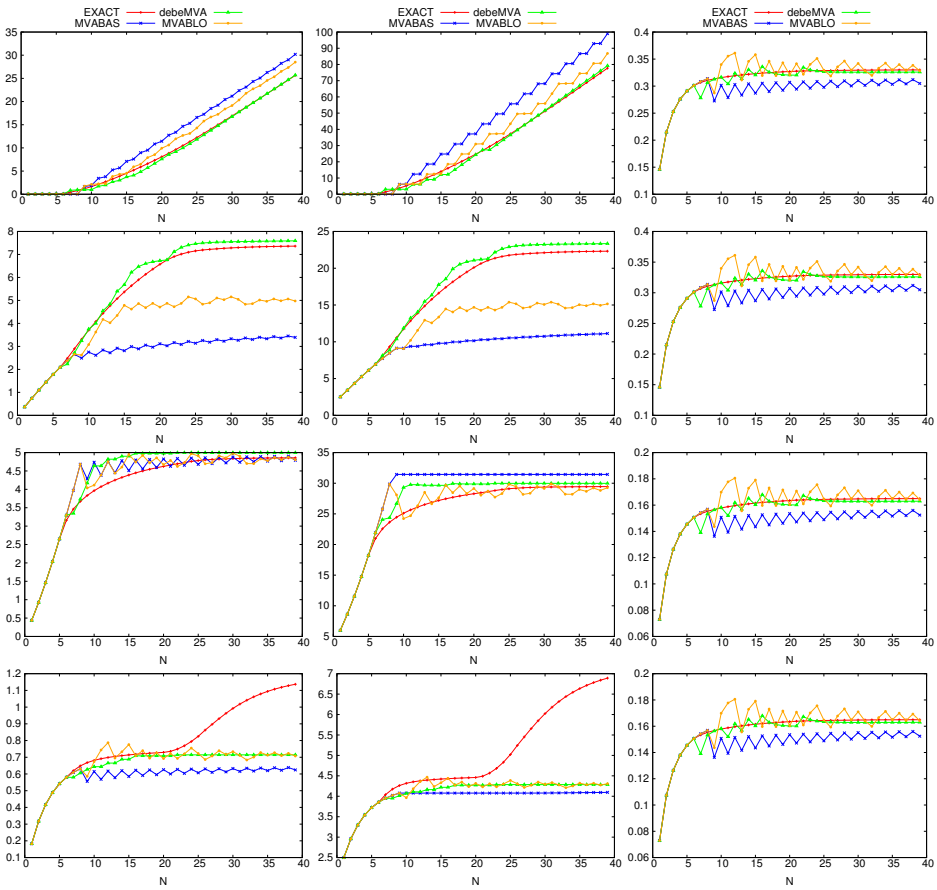
Figure 6: Network 46: node 0 in first row, node 1 in second row, node 2 in third row.

difficult case with 5 nodes and up to 4 possibly simultaneous transitions. In this case there are more critical points when several nodes become full. Note in Fig. 8 the first threshold when two nodes can be simultaneously full ($N = 6$). The second threshold is when three nodes can be simultaneously full ($N = 11$). After the thresholds the prediction is quite good.



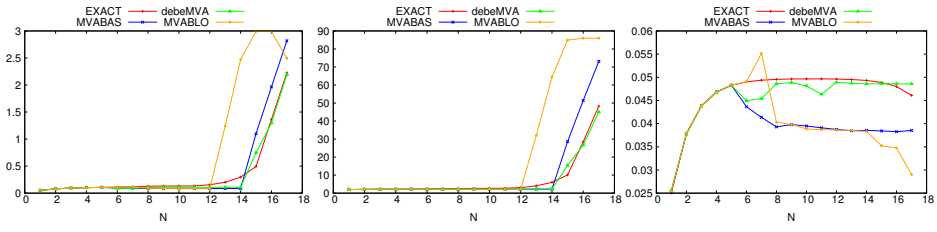
(a) Mean Population (b) Mean Residence (c) Throughput

Figure 8: Network 18 node 0



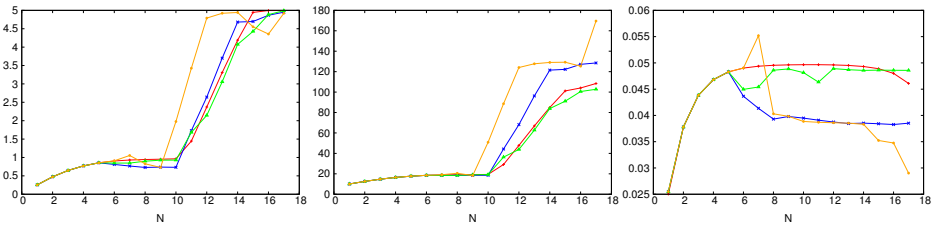
(a) Mean Population (b) Mean Residence (c) Throughput

Figure 7: Network 0: node 0 in first row, node 1 in second row, node 2 in third row, node 3 in forth row.



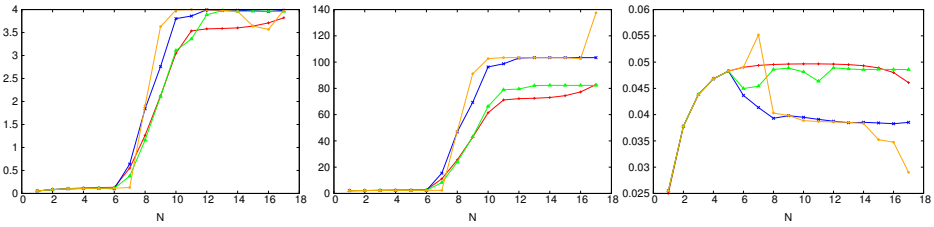
(a) Mean Population (b) Mean Residence (c) Throughput

Figure 9: Network 18 node 1



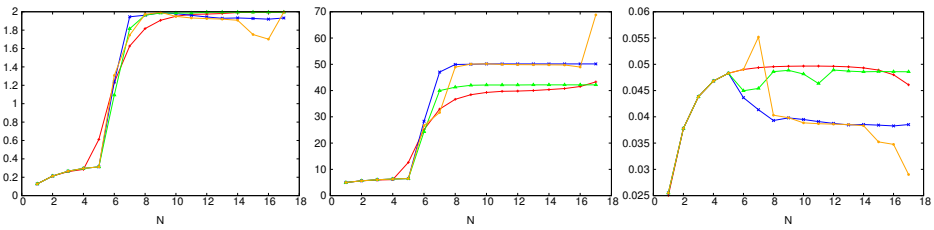
(a) Mean Population (b) Mean Residence (c) Throughput

Figure 10: Network 18 node 2



(a) Mean Population (b) Mean Residence (c) Throughput

Figure 11: Network 18 node 3



(a) Mean Population (b) Mean Residence (c) Throughput

Figure 12: Network 18 node 4

We conclude this section by comparing our method with [21]. Both algorithms modify the MVA equations. First, note that MVABLO does not consider delay centers. Moreover, it does not yield a good approximation and as soon as the congestion grows the results worse, especially for residence times and mean queue lengths. The results fluctuate between over- and under-estimating the performance metrics, so returning difficult error prediction (see e.g. throughput estimation in Figs. 6c and 7c and mean population estimation in Fig. 8a). Cyclic networks show the worst results.

As we introduced in Section 3, MVABLO recognizes saturation condition too late and does not capture blocking effective intensity. Indeed, it corrects mean populations always in the same measure: it simply cancels the arriving job ($z_j(k) = 0$). In conclusion, debeMVA differs from MVABLO in the following four points:

- the value for the arrival job: in [21], the arrival job function $z_i(k)$ is initialized to 1 and then simply set to 0 if a capacity violation is detected. In this paper, the AJ function is defined according to equations (2) and (3). These definitions allow to capture the dependence between nodes involved in a blocking occurrence. This is one of the main contributions of our method;

- the number of nodes involved by a blocking occurrence: according to Def. (1), a blocking can affect not only the two nodes directly involved, but also eventual other destinations of the blocked node. The proposed algorithm acts on all BIN nodes, while MVABLO acts just on the pair (s_BIN, f_BIN). As stated in previous section, our algorithm acts on all pairs (s_BIN, f_BIN) and (s_BIN, d_BIN) for any d_BIN.

- the condition under which the modification starts: in [21], the modifications start as soon as a mean population exceeds node capacity. As Akyildiz recognized, this is not enough to capture a blocking occurrence that can arise much early than the average node population approaches the node capacity. We propose an initialization phase and successfully overcome this limitation, see Section 4.2;

- the definition of the blocking time: in [21], the blocking time is defined as follows:

$$BT_i(k) \leftarrow BT_i(k) + p_{ij}\frac{e_i}{e_j}E[s_j] \tag{8}$$

  where e_i is the mean number of visits to node i. Equation (5) is quite different and the relative velocity factor ν_{ij} is another original contribution of our method for modeling the dependence between two nodes.

As we see in the next section, debeMVA greatly improves the approximation.

5. The results, error and cost analysis

The debeMVA algorithm has been tested on 56 networks that differ in number of nodes, topologies and parameters [28]. We considered cyclic networks and central server topologies. Cyclic topologies are particularly appropriate to stress the simultaneous transitions mechanism of BAS blocking. Note that in a cyclic network with M node up to $M-1$ nodes may be simultaneous blocked. As the results show, cyclic topologies are the most difficult ones to approximate, with the biggest number of iterations for convergence. On the other hand, central server topologies offer the possibility to investigate the effect of blocking even on nodes that are not directly involved in blocking (d_BIN in Def. (1)). Note that in all considered networks, we do not consider IS nodes. It is easy to be convinced that an IS node alleviates the negative impact of blocking: by definition it is an infinite capacity node. However, the case study model in Section 6.1 includes an IS node. Finally, we used different parameter values on the same topology to investigate their effect on the blocking phenomenon. As one can expect, the more difficult cases are when there is an unbalanced condition in the system,

e.g. some nodes are very fast while some other are very slow, or some nodes are never blocked while some other are blocked the most of the time.

We compare the results obtained by debeMVA with the results by MVABAS [23], MVABLO [21], and exact solution [1]. In [28], we show all results for each network, in terms of both performance indices and errors while in this section we show overall error and cost analysis.

First, we present statistics to evaluate the goodness of the proposed approximation and its computational cost. For each network, we evaluate the approximation error for each performance index and each node. In addition, given the iterative structure of the algorithm, we define e as the average error over the population as follows

$$e = \sum_{k=1}^{N} \frac{err_k}{N} \tag{9}$$

where err_k is the relative error for population k. To the aim of an overall evaluation of the method goodness, we define global average error, the minimum and the maximum average errors and the worst case error as follows:

- $\overline{e}$ is the global mean error on all relative errors $e_{i,j}$, for node i and performance index j, where j may be mean population n, mean response time t and mean throughput X

$$\overline{e} = \sum_{i=1}^{M} \sum_{j \in \{X,t,n\}} \frac{e_{i,j}}{3 \cdot M} \tag{10}$$

- e_m is the minimum relative error over all nodes and performance metrics

$$e_m = \min_{i \in [1,M],\, j \in \{X,t,n\}} e_{i,j} \tag{11}$$

- e_M is the maximum relative error over all nodes and performance metrics

$$e_M = \max_{i \in [1,M],\, j \in \{X,t,n\}} e_{i,j} \tag{12}$$

- err_M denotes the worst case, that is the maximum relative error over all computed errors

$$err_M = \max_{i \in [1,M],\, j \in \{X,t,n\},\, k \in [1,N]} err_{i,j,k} \tag{13}$$

Note that $\overline{e}$, e_m, e_M, are errors averaged over all possible population values, while $err_{i,j,k}$ and err_M are errors for just one population value k.

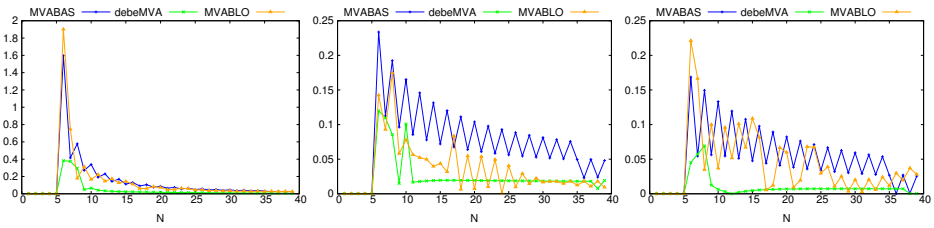
In Table 4, we see that debeMVA outperforms both previous methods for global average error $\overline{e}$ and for both the minimum e_m and maximum e_M average errors. In the last row of the table, we show also the maximum error err_M, that represents the worst case. This is greater than in the previous MVABAS, but still remains significantly lower than in MVABLO.

	MVABAS	debeMVA	MVABLO
$\overline{e}$	0.08	0.04	0.12
e_m	$1.47 \cdot 10^{-8}$	$1.47 \cdot 10^{-8}$	$2.90 \cdot 10^{-5}$
e_M	0.65	0.46	2.67
err_M	2.81	3.98	24.95

Table 4: Error statistics for 56 networks

It is worth noting that not only the average errors are improved, but most importantly the error trend is greatly improved by debeMVA as compared to previous algorithms. In the following, we show relative errors for the three networks presented in Section 4.4. To save space, we present errors for some performance measures, while the complete error analysis is in [28].

In Figure 13, we show the relative errors for performance metrics of node 0 and node 2 in Network 46, presented in Figure 6 first and third rows respectively. In Figure 14, we show the relative errors for mean population of node 1, node 2, and node 3 in Network 0, presented in Figure 7, first column. These populations are very difficult to approximate, as we have described in Section 4.4. Just the error for node 2 is slightly higher with debeMVA, but its trend is less erratic. Finally, in Figure 15, we consider Network 18 and show the relative errors for mean population of node 0 (Fig.8a), for response time of node 1 (Fig.9b) and for throughput that is the same for all nodes (Fig.8c-12c). Cyclic networks are the most difficult ones to approximate and debeMVA dramatically improves the results.

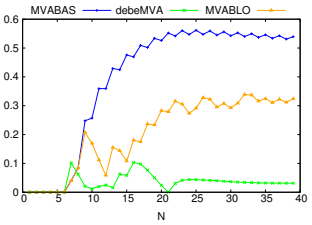


(a) Node 0 mean population (b) Node 0 throughput (c) Node 2 mean population

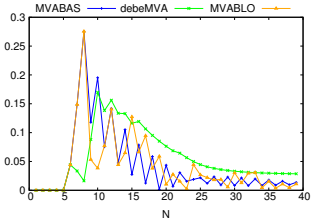
Figure 13: Relative errors for Network 46

It is easy to be convinced of the following characteristics: (i) generally all errors are lower than for the other two algorithms; (ii) beyond an initial phase to assess the approximation as soon as blocking occurs in one or more than one nodes, all trends are more stable and tend towards zero compared with the previous solutions, this is an important and promising characteristic of our solution.

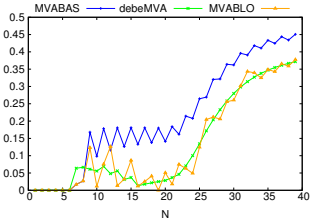
For the computational cost definition we consider both the execution time and the extra computation required with respect to traditional MVA. We measure the extra cost by the number θ of capacity violations (see Section 4.2) generated by the approximation. Note that this is also the number of extra iterations required by our method to converge. Note that all the considered cases converge to a solution. We define:



(a) Node 1 mean population

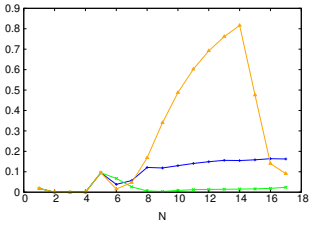


(b) Node 2 mean population

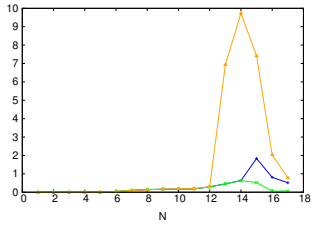


(c) Node 3 mean population

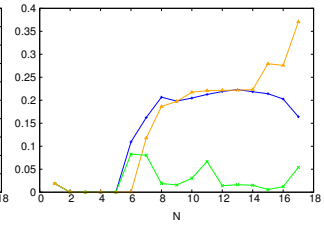
Figure 14: Relative errors for Network 0



(a) Node 0 mean population



(b) Node 1 response time



(c) Node throughput

Figure 15: Relative errors for Network 18

- θ: number of capacity violation for a given network;
- $\overline{\theta}$: mean number of capacity violation (ncv) for a set of networks;
- σ_θ: standard deviation of ncv for a set of networks;
- ε: execution time for a single network;
- $\overline{\varepsilon}$, σ_ε, ε_m, ε_M: mean, standard deviation, minimum and maximum execution time respectively for a set of networks.

	EXACT	MVABAS	debeMVA	MVABLO
$\overline{\theta}$	–	58.12	323.03	61.25
σ_θ	–	98.01	876.48	92.76
$\overline{\varepsilon}$ **(ms)**	173.35	13.8	13.39	13.25
σ_ε **(ms)**	597.14	1.32	1.33	1.27
ε_m **(ms)**	22.0	11.0	11.0	11.0
ε_M **(ms)**	3376.0	16.0	17.0	16.0

Table 5: Execution time and convergence statistics for 56 networks

In Table 5, the reader can see that the number of violations $\overline{\theta}$ is one order greater than the number in both two previous methods, with greater standard deviation too. This means that our method catches blocking occurrence more frequently than past methods. Nevertheless, the execution time is comparable. In other words, our method works better at the same cost. In [28], the interested reader can find errors and costs for each network of the experimentation.

6. Case study

Healthcare is one of the most critical and important sector of the social life. The aim of the World Health Organization is ambitious [29]: "the attainment by all people of the highest possible level of health", where "Health is a state of complete physical, mental and social well-being and not merely the absence of disease or infirmity." The role of IT in healthcare is becoming more and more significant, as numerous specific conferences and journals attest.

We consider planning decisions in healthcare [30]. The choice of this case study is also due to the importance of ease of use tools for this field. Indeed, the proposed method can be easily employed even by non-experts. Note that queueing networks with BAS blocking can well represent a network of care structures in which the capacity to admit patients is finite. Moreover, the "after service" blocking exactly corresponds to the conclusion of therapy in the current structure. In that case, when a patient hospitalized cannot be admitted in the next appropriate structure, he/she blocks the current hospital bed. The health system lose both in care efficiency (for the delay in the appropriate therapy) and in financial perspective. As a consequence, controlling the congestion and planning the appropriate size for the structures are important goals.

As a case study, we consider the Philadelphia mental health system that was the forefront of the health system rearrangement [7]. In this case, the downsizing and the rearrangement of mental health institutions is to provide mentally ill patients with a "continuum care network of facilities with various levels of structure and support". As introduced above, BAS blocking conforms to model patients that find full the structures where they were referred and are forced to wait at their current facilities. These patients "block" beds in the current structures, preventing their utilization by potential patients requiring care at these facilities. This behaviour is quite well captured by BAS

blocking and by the simultaneous transitions mechanism. Fig.16 describes the different structures and the patients flows. Next section introduces the model for this system.

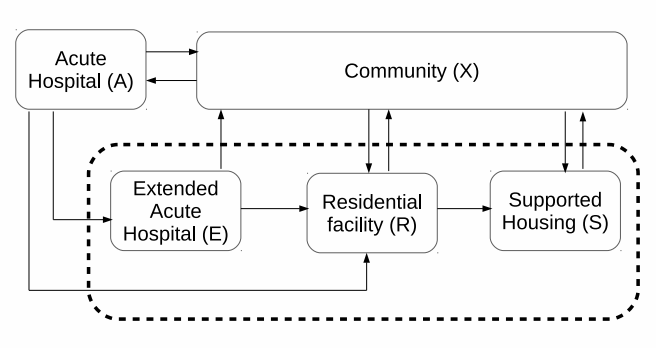


Figure 16: Structures and patients flows

6.1. System model and baseline performance

We model the considered healthcare system with the closed queueing network with finite capacity and BAS blocking, see Fig.17. We consider a closed model to analyze the behaviour in case of congestion and assume $N = 1700$ patients. Each center represents a care structure with a finite number of beds. Note that, for the lack of specific parameters for the acute hospital center A in [7], we collapse the two structures A and X in a unique IS center denoted by X.

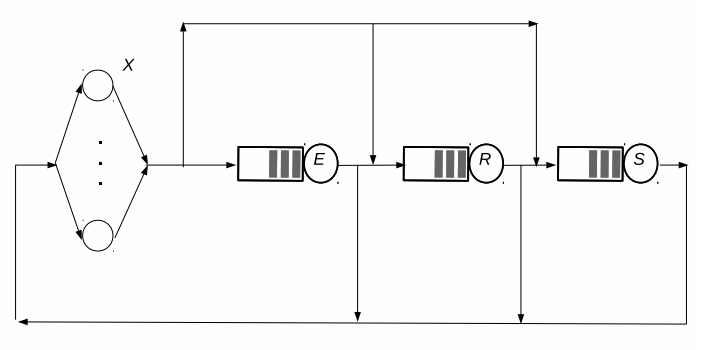


Figure 17: Queueing network model

By assuming congestion, we can model a structure as a finite capacity single server, but with an ”aggregate” service rate to take into account the ”service” of each bed. In [28], we derive the model parameters starting from values measured from the real system [7]. In particular, the routing matrix is derived from external or internal arrival rates to the care structures. Tables 6 and 7 show the parameters values.

	X	E	R	S
X	0	0.223	0.664	0.113
E	0.748	0	0.252	0
R	0.943	0	0	0.057
S	1.0	0	0	0

(a) Routing matrix

d_i **(days)**		
E	**R**	**S**
60	893	2500

(b) Mean service terapy time for each structure

B_i		
E	**R**	**S**
64	1206	416

(c) Maximum capacity structure (number of beds)

Table 6: Parameters observed from the real system

$E[s_i]$ (days)			
X	**E**	**R**	**S**
0.331	0.94	0.74	6.0

Table 7: Mean service time for queueing model

Without any predictive analysis, one could naively think that the structure R is the most congested one since this is selected by the most patients. This is confirmed by the routing probability value in Table 6a. This could be misunderstood by leading to the R upgrade. In the following section, we show that a different direction is most beneficial. Low computational cost of our method is appealing to study several alternative solutions so to take planning decisions opportunely.

First, we present the results obtained for the baseline system under congestion with N=1700 users/patients, that is a value greater than the effective capacities of the care structures. Moreover, it is worth noting that we use the total demand (called referral arrival rates in [7]) for care facilities to consider the system with no reneging from queues. In this way, the results represent the "response" of the baseline system to the original demand. The objective is to compute the mean residence time of the patients in each structure.

Table 8 shows the mean residence time and the mean population in each structure. Note that the most patients are in the R structure.

$E[t_i]$ (days)			
X	**E**	**R**	**S**
17.835431	265.940554	1546.287962	2495.999970

(a) Mean residence time for each care structure

$E[n_i]$			
X	**E**	**R**	**S**
17.708694	63.679830	1202.611481	415.999995

(b) Mean population for each care structure

Table 8: Mean population and residence time computed using debeMVA

By assuming that the residence time $E[t_i]$ is not lesser than the mean treatment time d_i, the mean blocking time in a structure i for each patient can be evaluated as follows:

$$w_i = E[t_i] - d_i \tag{14}$$

Table 9 shows the mean blocking time w_i for each structure. Note that the structure S is omitted since it cannot be blocked. R is the structure that experiment the most blocking time. In the next section, we present a capacity planning study to select the most convenient structure to be upgraded.

Estimated blocking time (days)		
X	**E**	**R**
17.504431	205.940554	653.287962

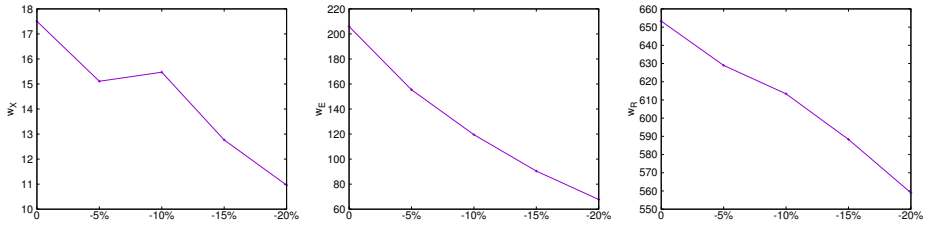
Table 9: Mean blocking time for each care structure

Finally, as a sensitivity analysis we consider the effect of increasing admission rate in the structure Extended Acute Hospital E, for the benefit of less intensive structures R and S. Since the parameters are derived from the real system [7], we modify routing probabilities as follows: 1) we decrease of 5% for four times the routing probabilities to both less intensive care structures R and S; 2) we correspondently increase p_XE, so to have consistent routing probabilities. In this way, we keep the same proportion of use between the two care structures R and S as suggested by the real case. Table 10 shows the first row of the routing matrix for the four steps. The other rows are the same as in Table 6a.

	X	**E**	**R**	**S**
-5%: X	0	0.262	0.631	0.107
-10%: X	0	0.301	0.598	0.102
-15%: X	0	0.340	0.564	0.096
-20%: X	0	0.378	0.531	0.090

Table 10: Increasing admission rate: first row of routing matrix

Fig. 18 shows that blocking time decreases in each structure as the admission rate increases for the Extended Acute Hospital. While this is not surprising for structure E, having reduced the workload on its destination structure R (see point 1 above), it seems unexpected for the other two structures X and R. We think that this behaviour is due to the fact that the Extended Acute Hospital has the fastest mean service terapy, while the structures Residential Facility and Supported Housing have the longest terapy times (see Table 6b). In other words, as admission rate grows, workload is moved to the fastest structure E, with the effect of alleviating blocking. This confirms also our intuition that node velocity is most influential on blocking than saturation itself (see initialization phase in Section 4.2, and the relative velocity factor ν_{ij} defined in eq. (6)). It is worth noting that several considerations should be taken into account to take decisions: care efficiency should be measured also in terms of individual well-being. It is well known that the place of care has its influence on health. The discussion of this interesting topic is out of the scope of this paper. We include this sensitivity analisys as a toy example to show the potentiality of an ease to use method. Starting from the reality of the healthcare system under study, a lot of interesting studies can be easily conducted.



(a) Blocking time for X (b) Blocking time for E (c) Blocking time for R

Figure 18: Estimated mean blocking time by increasing admission rate

6.2. Capacity planning

In this section, we consider the possibility to upgrade a structure and we use our efficient solution method to evaluate different alternatives. As an example, we consider an upgrade of extra beds just for one care facility. By assuming budget availability for other 20 beds, the aim of the study is the identification of the appropriate structure to upgrade with the extra beds from both the perspectives of the human rights and the finance [7].

Table 11, shows the parameter sets B_i and $E[s_i]$ for the baseline and all the alternatives. We denote with $Y+$ the enhanced structure; it is worth noting that the upgrade increases the structure capacity B_Y and as a consequence decreases the mean service time $E(s_Y)$.

B_i			
Model	**E**	**R**	**S**
Baseline	64	1206	416
E+	84	1206	416
R+	64	1226	416
S+	64	1206	436
$E[s_i]$			
Model	**E**	**R**	**S**
Baseline	0.94	0.74	6.0
E+	0.71	0.74	6.0
R+	0.94	0.72	6.0
S+	0.94	0.74	5.73

Table 11: Parameter sets for baseline model and all alternatives

Table 12 shows the results obtained for all the alternatives in terms of the mean residence time and the mean blocking time in each structure. The most improvement is obtained by upgrading the structure S: indeed the congestion in this structure can involve a cascade blocking in the structures X, E and R. By upgrading S, both the residence times and the blocking times of all the other structures are improved. On the contrary, if the upgrade was on the structure R, both the structures X and E improve their performance, while the blocking time in the structure R itself grows: indeed the greater beds availability produces more blocked patients waiting for care in S. Note that this is a counterintuitive result. Finally, if upgrade was on the structure E, just the blocking time for new patients in structure X improves, but the blocking time in structure E grows significantly, while in the structure R it is not improved. Note also that in these last two cases the improvements are less pronounced than in the upgrade of S. We remark again the importance of an easy and low cost method as debeMVA to conduct analysis of several alternatives for capacity planning.

In Table 13, we show the number of capacity violation detected for each alternatives (first row) and the relative execution time (second row). It is worth noting that number of violations collapses of 2 order of magnitude (first row) by just increasing of 20 a total capacity of 1686 beds. On the other hand, with so many more violations (baseline) the computational time is just one order of magnitude bigger (second row). In any case, it is easy to be convinced that the cost of evaluating several alternatives is quite negligible.

$E[t_i]$ (days)				
Model	**X**	**E**	**R**	**S**
Baseline	17,835431	265,940554	1546,287962	2495,99997
E+	1,815789	316,079355	1548,473325	2495,999755
R+	1,632431	234,159152	1574,300692	2495,999793
S+	1,47779	224,08174	1478,979632	2498,279585
Estimated blocking time (days)				
Model	**X**	**E**	**R**	**S**
Baseline	17,504431	205,940554	653,287962	–
E+	1,484789	256,079355	655,473325	–
R+	1,301431	174,159152	681,300692	–
S+	1,14679	164,08174	585,979632	–

Table 12: Mean residence time and estimated blocking time for each care structure

	Baseline	E+	R+	S+
$\boldsymbol{\theta}$	177316	5221	4563	4104
ε **(ms)**	128	67	65	63

Table 13: Execution time and convergence statistics for alternatives

We conclude this section by highlighting the main differences between our modeling approach and the approach in [7]. First we consider a closed model instead of an open one. This means that with real parameters for A, we could explicitly model the relapse represented by the backflows from the structures R and S towards the acute hospital A. Moreover, the authors consider "the waiting space to enter a particular station as infinite" (see Section 4.2 in [7]) so that in the steady state the number of patients waiting to enter a station can be bigger than the number of beds at the previous station. On the contrary, in our model each station may have finite capacity, so we can represent all the finite resources of the healthcare system. Moreover, since they applied the decomposition method, the effect of cascade blocking for more than two stations is not represented by their model.

An important difference between the two approaches regards the considered arrival rates. The authors consider the effective *admission* rates that not include the patients referred to the facilities but reneged from the waiting lists. Indeed, with the referral rates, the open model does not reach stability condition. So they cannot include the "true demand" in their analysis.

In this paper, we consider the referral rates to derive the routing probabilities [28], but by using a closed model we can investigate about congestion without incurring in the stability problem. Finally, it is worth noting that the proposed method is quite appropriate for several investigations since its easy and low-cost use (see Table 13).

7. Conclusions

In this paper, we have proposed an approximate MVA for queueing networks with BAS blocking. The method is based on the definition of functions to capture the inter-dependency among nodes involved in the blocking occurrence. So the non-separability condition of general blocking networks is captured and approximated in the mean performance metrics. We think this approach could be applied to different forms of dependence, so other non-separable networks could benefit of a similar approach. By investigating the nature of dependence, functions of node i should be opportunely defined including elements of node j representing the behaviour dependence of node i on node j. This is the essence of our approach, e.g. by means of AJ functions and BT_i function. The method is tested on 57 networks, different in topologies and characteristics so to include balanced and unbalanced cases. The experimentation shows good results and a significant improvement with respect to previous algorithms. We evaluate mean global errors, minimum and maximum errors and the worst case. Moreover we consider the computational cost and specifically the overhead due to extra iterations to approximate the mean node population for a node involved in blocking.

To showcase the method at work, we consider healthcare systems which are a critical and important sector of social life. In this case, capacity planning studies are very important both for care efficiency and economical perspective. In particular, we consider the Philadelphia mental health system that was the forefront of the health system rearrangement. The proposed method is particularly suitable for non-experts that have to take decisions in order to rearrange structures. We consider the case study in [7] and obtain the same results, but at a lower computational cost and with a more flexible approach that open to further investigations. We show that one can easily investigate all cases of upgrading different structures and select the most convenient and efficient solution.

Several future directions can be investigated. Some results are not still good and show that the method needs to be further improved. The use of non-exponential distributions could be investigated, the inclusion of blocking on a subnetwork would be an interesting feature to model admission control mechanisms. Different blocking mechanisms could be considered and a similar approach could be investigated for different dependencies, e.g. load-dependent routing.

Acknowledgments

The authors would like to thank the reviewers for their suggestions that really improved the quality of paper.

REFERENCES

[1] S. Balsamo, V. de Nitto Personé, R. Onvural., Analysis of queueing networks with blocking., Kluwer Academic.

[2] R. Onvural, Survey of closed queueing networks with blocking., ACM Computing Surveys 22 (2) (1990) 83–121.

[3] R. Onvural, Special issue on queueing networks with finite capacity., Perform. Eval. 17 (3).

[4] H. Perros, Queueing networks with blocking., Oxford University Press.

[5] S. Balsamo, V. de Nitto Personé, P. Inverardi., A review on queueing network models with finite capacity queues for software architectures performance prediction., Perform. Eval. 51 (2-4) (2003) 269–288.

[6] L. Lu, L. Cherkasova, V. de Nitto Persone, N. Mi, E. Smirni, AWAIT: Efficient overload management for busy multi-tier web services under bursty workloads, In proc. of ICWE 2010, Web Engineering, 10th International Conference, Vienna, Austria 46 (2010) 455–477.

[7] N. Koizumi, E. Kuno, T. Smith, Modeling patient flows using a queuing network with blocking, Health Care Management Science 8 (2005) 49–60.

[8] M. Reiser, S. Lavenberg, Mean value analysis of closed multichain queueing networks, J. ACM (1980) 313–322.

[9] M. Jain, S. Maheshwari, K. Baghel, Queueing network modelling of flexible manufacturing system using mean value analysis, Applied Mathematical Modelling 32 (5) (2008) 700–711.

[10] X. Guoqing, C. Hongzhao, W. Yuanhui, Analysis of aircraft sortie generation process using a closed queuing network model, In Proc. of 2010 International Conference on Intelligent Computation Technology and Automation (ICICTA) (2010) 643–646.

[11] E. Winands, I. Adan, G. van Houtum, Mean value analysis for polling systems, Queueing Systems 54 (1) (2006) 35–44.

[12] D. Petriu, C. Woodside, Approximate mean value analysis based on markov chain aggregation by composition, Linear Algebra and its Applications 386 (2004) 335–358.

[13] A. Bogardi-Meszoly, T. Levendovszky, A novel algorithm for performance prediction of web-based software systems., Perform. Eval. 68 (2011) 45–57.

[14] A. Kattepur, M. Nambiar, Modeling of multi-tiered web applications with varying service demands, In Proc. of IEEE International Parallel & Distributed Processing Symposium IPDPS 2015 Workshops (2015) 415–424.

[15] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, A. Tantawi, An analytical model for multi-tier internet services and its applications, An Analytical Model for Multi-tier Internet Services and its Applications (2005) 291–302.

[16] Q. Zhang, L. Cherkasova, E. Smirni, A regression-based analytic model for dynamic resource provisioning of multi-tier applications, In Proc. of the 4th ICAC Conference (2007) 27.

[17] L. Zhang, D. Down, Approximate mean value analysis for multi-core systems, In Proc. of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS) (2015) 643–646.

[18] G. Casale, E. Smirni, MAP-AMVA: Approximate mean value analysis of bursty systems, In Proc. of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (2009) 409–418.

[19] M. Tribastone, Approximate mean value analysis of process algebra models, In Proc. of IEEE19th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS) (2011) 369–378.

[20] M. Sereno, Mean value analysis of product form solution queueing networks with repetitive service blocking, Perform. Eval. 36-37 (1999) 19–33.

[21] I. F. Akyildiz, Mean value analysis of blocking queueing networks, IEEE Trans. On Software Eng. 14 (4) (1988) 418–428.

[22] M. Yuzukirmizi, Performance evaluation of closed queueing networks with limited capacities, Turkish J. Eng. Env. Sci. (2006) 269–283.

[23] V. De Nitto Personé, A. Di Lonardo, An Approximate Mean Value Analysis Approach for System Management and Overload Control, in: Computer Performance Engineering, Vol. 8168 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 288–299.

[24] T. Yamadaa, N. Mizuharab, H. Yamamotoc, M. Matsuib, A performance evaluation of disassembly systems with reverse blocking, Computers & Industrial Engineering Intelligent Manufacturing and Logistics (2009) 1113–1125.

[25] A. Lazar, G. Robertazzi, The geometry of lattices for markovian queueing networks, Perform. Eval. 6 (1986) 85–86.

[26] S. Lavenberg, M. Reiser, Stationary state probabilities of arrival instants for closed queueing networks with multiple types of customers, J. of Applied Probability 17 (4) (Dec. 1980) 1048–1061.

[27] K. Sevcik, I. Mitrani, The distribution of queueing network states at input and output instants, J. of ACM 28 (2) (Apr. 1981) 358–371.

[28] V. De Nitto Personé, A. Di Lonardo, MVABAS v2: experimentation networks and results. url: https://art.torvergata.it/retrieve/handle/2108/197911/391720/mvabastechreport.pdf, Research Report, RR-18.14.

[29] WHO, Basic documents url: http://apps.who.int/gb/bd/pdf/bd48/basic-documents-48th-edition-en.pdf, Forty-eighth edition.

[30] P. Hulshof, N. Kortbeek, R. Boucherie, E. Hans, P. Bakker, Taxonomic classification of planning decisions in health care: a structured review of the state of the art in or/ms, Health Systems 1 (2) (2012) 129–175.