Chapter 2
SMVA: A Stable Mean Value Analysis Algorithm for Closed Systems with Load-Dependent Queues

Lei Zhang and Douglas G. Down

2.1 Introduction

The Mean Value Analysis (MVA) algorithm [16] is an efficient solution for steady-state analysis of queueing networks. However, it relies on product-form assumptions, which can be violated by common features introduced in modern computer systems, e.g., simultaneous resource possession, locking behaviours, priority scheduling, high service demand variability, and process synchronization (see Chapter 15 in [14]). An approximate solution is to reduce a non-product-form network by using Flow-Equivalent Servers (FESs) [7]. An FES is load dependent, whose service rate with n jobs present is equal to the observed throughput of the original network with n jobs. The performance model can then be analysed using the load-dependent MVA algorithm [15].

Unfortunately, the load-dependent MVA algorithm suffers from numerical instability issues [15, 16]. The underlying reason is that the computation of state probabilities can yield negative results when the utilization is close to one. Consequently, negative values of mean performance measures (i.e., response times and throughputs) can be produced. Static and dynamic scaling techniques are potential approaches to cope with precision limits, but they are complicated to implement. In addition, Casale and Serazzi [4] show that they do not work in general, as the mean queue length computations are not affected. To the best of our knowledge,

L. Zhang
Department of Computer Science, Ryerson University, Toronto, ON, Canada
e-mail: leizhang@ryerson.ca

D. G. Down (✉)
Department of Computing and Software, McMaster University, Hamilton, ON, Canada
e-mail: downd@mcmaster.ca

A. Puliafito, K. S. Trivedi (eds.), *Systems Modeling: Methodologies and Tools*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-319-92378-9_2

the literature is lacking efficient solutions for the numerical instability of load-dependent MVA.

In this paper, we propose a Stable MVA (SMVA) algorithm for closed networks with load-dependent queues (the initial idea was presented in [21]). The main contributions of this paper include: (1) the SMVA algorithm, which is an efficient approximate solution for closed networks with load-dependent queues, and (2) an extended multi-class model used to determine class-level performance metrics.

This paper is structured as follows. Section 2.2 introduces the required background. Section 2.3 provides a review of solutions proposed in the literature for the numerical instability. We then present SMVA in Sect. 2.4. In Sect. 2.5, the results from SMVA are compared with other MVA algorithms in two case studies. Section 2.6 gives the multi-class SMVA algorithm. This paper ends with Sect. 2.7, in which we give a summary of the pros and cons of SMVA.

2.2 Background

The exact MVA algorithm for closed networks with load-dependent queues has numerical instability issues. It may exhibit numerical difficulties under heavy load conditions which eventually result in unreasonable results, such as negative throughputs, response times, and queue lengths. The numerical problem is that the probability of a resource being idle is calculated in every iteration of the load-dependent MVA algorithm. The calculation is as follows:

$$P_m(0|n) = 1 - \sum_{i=1}^{n} P_m(i|n), \tag{2.1}$$

where $P_m(i|n)$ is the probability that i jobs are at the mth resource when a total of n jobs are in the system. When the utilization is close to one, (2.1) can yield negative values, and those errors propagate as the MVA algorithm iterates. Subsequently, other calculations which have direct or indirect dependence on (2.1) may result in negative values, such as mean response times and throughputs, which do not make any physical sense.

2.3 Related Work

Chandy and Sauer [6] provide the initial reports of the numerical instability of the MVA algorithm with load-dependent queues. Reiser [15] confirms this issue. To replace (2.1) in a single-class model, he proposes a new calculation of $P_m(0|n)$ evaluated by $P_m(0|n-1)$ and the throughput $X^{[m]}(n)$ in an m-complement system, which is defined as a queueing system without the mth queue and all other parameters remaining the same. However, the expense of the evaluation grows exponentially as the number of load-dependent queues increases.

Tucci and Sauer [20], and Hoyme et al. [11] independently propose two similar tree-structured MVA algorithms, which are invulnerable to numerical instability. The main idea is to build a tree data structure, where queues are leaves. The internal nodes are intermediate functions, resulting from convolving all queue functions in the subtree with the internal node as the root. For dense queueing networks, tree MVA algorithms can give even worse performance than the original MVA algorithms whose complexities grow linearly, but they are efficient when customers visit only a small number of queues.

Casale et al. [5] suggest an approximate MVA algorithm (QD-MVA) for queue-dependent stations in a multi-class setting. Its computational cost is $O(MC)$ for a model with M queues and C classes. However, it may not converge in some instances. Moreover, it relies on queue-dependent functions to analyse queue-dependent service times, which introduces excessive computational requirements. They show that the QD-MVA algorithm has very good accuracy for the estimation of mean queue lengths, but the results from QD-MVA on other performance metrics, such as mean response times and system throughput, are not provided.

In the literature, Seidmann's approximation [18] is also widely used to address MVA's numerical issues [8, 9, 13]. The basic idea is to replace a multi-server queue with k servers by two tandem servers. The first one is a single server queue with service demand D/k, where D is one server's service demand. The second one is a pure delay server with service demand $D \cdot (k-1)/k$. In practice, Seidmann's approximation can yield noticeable errors under intermediate loads, but it has the same time and space complexities as the original MVA algorithm. However, Seidmann's approximation assumes that the servers in the multi-server queue are load independent. Such an approximation may not be realistic when the FES technique is employed.

To address numerical issues, Casale [3] introduces the Conditional MVA (CMVA) algorithm, which avoids the computations of the state probabilities, and as a consequence, overcomes the limitation. Although the CMVA algorithm is an exact solution, its time and space complexities grow much faster than the original MVA algorithm. Given M is the number of queues, and N is the number of jobs, the time and space complexities for the original MVA algorithm only grow as $O(MN)$, while those for the CMVA algorithm grow as $O(MN^{L+1})$, where L is the number of load-dependent queues in the system . This may cause significant time and memory issues for the computation when N or L is large, which is very common in performance evaluation for stress tests.

2.4 Stable Mean Value Analysis

We study a closed queueing network with N jobs and M queues, and we focus on a generic load-dependent queue with service demand $D_m(n_m)$, where m is the index of the queue, and n_m is the number of jobs at the queue (with $n = \sum_{m=1}^{M} n_m$ where

$m = 1, \ldots, M$ and $n = 1, \ldots, N$). Here, we assume that the service demand of the load-dependent queue becomes a constant beyond some $\bar{N}_m$, i.e., there exists a finite $\bar{N}_m$ such that $D_m(n_m) = D_m(\bar{N}_m)$ for all $n_m \geq \bar{N}_m$. This assumption is reasonable for many systems, in particular when $D_m(n_m)$ becomes sufficiently close to $D_m(\bar{N}_m)$.

The basic idea of the SMVA algorithm is inspired by Seidmann's approximation, replacing the load-dependent queue with two tandem servers. The first is a load-independent (LI) queue with service demand $D^q_m = D_m(\bar{N}_m)$. The second is a load-dependent (LD) delay centre with service demand

$$D^d_m(n_m) = \begin{cases} n_m D_m(n_m) - D_m(\bar{N}_m), & \text{if } n_m < \bar{N}_m \\ (\bar{N}_m - 1) D_m(\bar{N}_m), & \text{if } n_m \geq \bar{N}_m. \end{cases} \tag{2.2}$$

To make sure the service demands in (2.2) are positive, we assume that $n_m D(n_m) \geq D(\bar{N}_m)$, for $n_m < \bar{N}_m$. In multi-core computer systems, it is a common assumption that $D_m(n_m)$ decreases as n_m increases, so $D_m(n_m) > D_m(\bar{N}_m)$ when $n_m < \bar{N}_m$ and $n_m D_m(n_m) \geq D_m(\bar{N}_m)$ holds. Although the delay centre is load dependent, there is no need to calculate its state probabilities because it does not have a queue. As a result, the SMVA algorithm is numerically stable.

Under light load, the two tandem servers behave as a server which has service demand $D_m(n_m)$. If n_m jobs are being served and no jobs are waiting in the queue, the time spent by a job in the approximating node is $D_m(\bar{N}_m) + n_m D_m(n_m) - D_m(\bar{N}_m) = n_m D_m(n_m)$. If there are jobs waiting in the first queue, the time spent by a job in the approximating node is dominated by the time spent at the first queue. The node behaves as a server which has service demand $D_m(\bar{N}_m)$. As a result, this approximation should perform well for both light and heavy loads. Note that SMVA is identical to Seidmann's approximation when $n_m D_m(n_m) = D_m(1)$, for $n_m \leq \bar{N}_m$.

Once we finish the service demand parameterization, the mean response times at the load-independent queue in the approximating network with n jobs can be computed by the arrival theorem [12, 19]:

$$R^q_m(n) = D^q_m[1 + Q_m(n - 1)], \tag{2.3}$$

where $Q_m(n-1)$ is the mean queue length at the mth queue with $n-1$ jobs in the network.

To compute the mean response time at the delay centre, we need to estimate the mean number of jobs, because its service demand is load dependent. We employ the Bard-Schweitzer approximation [1, 17] to estimate the mean number of jobs at the delay centre. The Bard-Schweitzer approximation is based on the following idea: The number of jobs at each queue increases proportionately as the total number of jobs increases in the network. Mathematically:

$$\frac{Q_m(n-1)}{Q_m(n)} = \frac{n-1}{n}. \tag{2.4}$$

There are two things that we need to clarify here: (1) We use the term “mean number of jobs” rather than “mean queue length”, because it is a pure delay centre, and it has no jobs waiting for service. (2) When we mention the mean number of jobs, we refer to the actual mean number of jobs of the original network instead of those of the approximating network. Let $Q_m^o(n-1)$ be the mean number of jobs at the mth queue when there are $n-1$ jobs in the network, and $Q_m^e(n)$ be the estimated mean number of jobs at the mth queue when there are n jobs in the network. We can rewrite (2.4) as:

$$Q_m^e(n) = \begin{cases} 1, & \text{if } n = 1, \\ \dfrac{n}{n-1} Q_m^o(n-1), & \text{if } n > 1. \end{cases}$$

Then, we can compute the mean response times at the delay centre as $R_m^d(n) = D_m^d(\lceil Q_m^e(n) \rceil)$. The ceiling function ensures that the index of the load-dependent service demands starts from one, rather than zero.

The system throughput is calculated using Little’s Law:

$$X(n) = n / \left\{ Z + \sum_{m=1}^{M} [R_m^q(n) + R_m^d(n)] \right\}, \tag{2.5}$$

where Z is the mean think time. To compute the mean queue length at the mth queue in the approximating network, we just continue applying Little’s Law: $Q_m^a(n) = X(n) \cdot R_m^q(n)$. The mean queue length at the mth queue in the original network is:

$$Q_m^o(n) = X(n) \cdot [R_m^q(n) + R_m^d(n)].$$

Algorithm 1 illustrates the single-class SMVA algorithm in detail. SMVA has two features: (1) SMVA is numerically stable, because it avoids the calculation of stationary probabilities at load-dependent queues; (2) SMVA is efficient, because its time and space complexities are both $O(MN)$.

There are two things that we would like to highlight in Algorithm 1. Firstly, we assume that all queues are load dependent in Algorithm 1. If the mth queue is load independent, we can simply set $D_m^q = D_m$ and $D_m^d = 0$, and Algorithm 1 is still applicable. Secondly, we do not check whether $Q_m^e(n)$ and $Q_m^o(n)$ converge to each other in SMVA, because we are iterating over n and we do not guess the initial values of $Q_m^e(n)$ (the Bard-Schweitzer approximation has both of them). In the Appendix, we propose an alternative SMVA algorithm which has a comparison between $Q_m^e(n)$ and $Q_m^o(n)$.

Algorithm 1 The single-class SMVA algorithm

```
Input:
$Z, M, N, D_m(n), \bar{N}_m$
Output:
$Q_m^o(N), X(N), R(N)$
Condition:
$nD_m(n) \geq D_m(\bar{N}_m), \forall n, m$
Initialization:
$Q_m^a(0) = 0, \forall m = 1, \ldots, M$
Iteration:
  for $m = 1 \to M$ do
    for $n = 1 \to N$ do
      $D_m^q = D_m(\bar{N}_m)$
      $D_m^d(n) = \begin{cases} nD_m(n) - D_m(\bar{N}_m), & \text{if } n < \bar{N}_m \\ (\bar{N}_m - 1)D_m(\bar{N}_m), & \text{if } n \geq \bar{N}_m \end{cases}$
    end for
  end for
  for $n = 1 \to N$ do
    for $m = 1 \to M$ do
      if $n = 1$ then
        $Q_m^e(n) = 1$
      else
        $Q_m^e(n) = \frac{n}{n-1} Q_m^o(n-1)$
      end if
      $R_m^q(n) = D_m^q[1 + Q_m^a(n-1)]$
      $R_m^d(n) = D_m^d(\lceil Q_m^e(n) \rceil)$
    end for
    $X(n) = n/\{Z + \sum_{m=1}^{M}[R_m^q(n) + R_m^d(n)]\}$
    for $m = 1 \to M$ do
      $Q_m^a(n) = X(n) \cdot R_m^q(n)$
      $Q_m^o(n) = X(n) \cdot [R_m^q(n) + R_m^d(n)]$
    end for
  end for
  $R(N) = \sum_{m=1}^{M}[R_m^q(N) + R_m^d(N)]$
```

2.5 Experimental Results

In order to verify the accuracy and the efficiency of the SMVA algorithm, we compare the results of the SMVA algorithm, the CMVA algorithm, and Seidmann's approximation in two different closed queueing networks. The first one is a closed network with one generic load-dependent queue (FES), and the second one is a closed network with two FESs. To generate the input parameters—service demands—for these two queueing networks, we set up a testbed on an Intel i7-2600 quad-core computer with 8 GB memory, 1 TB hard drive, and Ubuntu 12.04.3 LTS. We employ JBoss 3.2.7 as the application server, MySQL 5.1.70 as the database server, and TPC-W [10] to generate the workload. TPC-W can simulate three

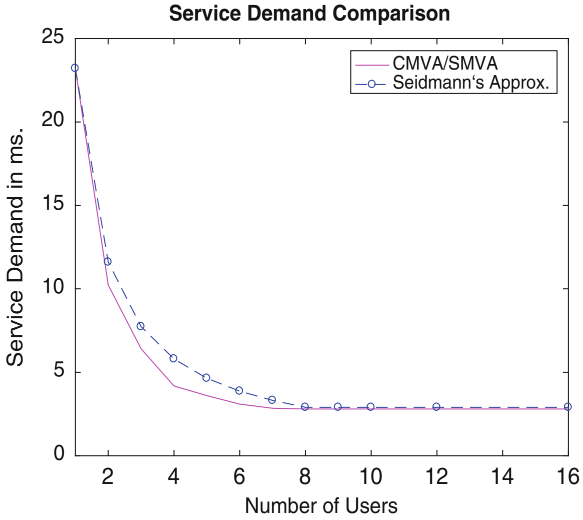


Fig. 2.1 Service demands for browsing

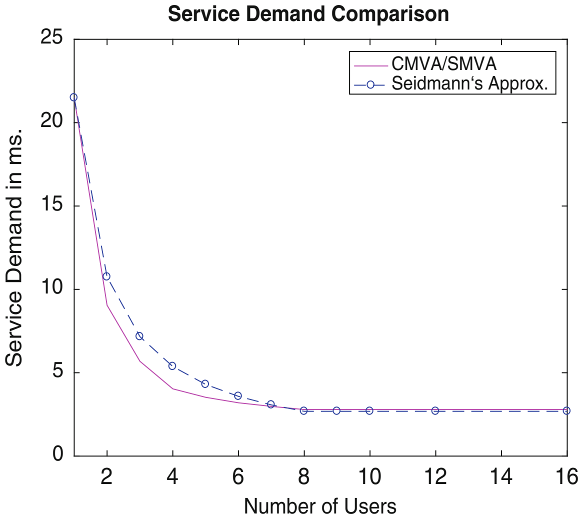


Fig. 2.2 Service demands for shopping

workloads for an e-commerce environment—browsing, shopping, and ordering. We choose the first two workloads in our tests, and plot their service demands in Figs. 2.1 and 2.2.

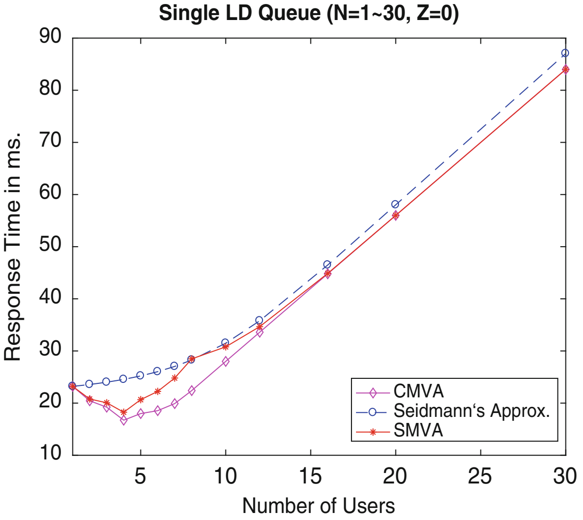
2.5.1 One Load-Dependent Queue

For the first network, we aggregate and model the computer system by an FES. We then run the browsing workloads to obtain the system throughputs, and calculate the service demands of the FES to parameterize the MVA algorithms (as shown in Fig. 2.1). As can be seen in the figure, the service demand curve adopted by Seidmann's approximation can only address the ideal case of a load-dependent server, where $D(n) = D(1)/n$ for $n \leq 8$. By overestimating the service demands in such a case, the outcomes of Seidmann's approximation are conservative in terms of performance metrics, but this may not be true in general.

To test the accuracy of SMVA under different loads, we vary the number of users and the mean think time in the system. Both the mean response time and the throughput are compared for the three candidate MVA algorithms. Three sets of results are presented. The results of the first set are presented in Figs. 2.3 and 2.4, where N ranges from 1 to 30, and $Z = 0$. The results of the second set are presented in Figs. 2.5 and 2.6, where N ranges from 1 to 300, and $Z = 0.7$ s. The results of the third set are presented in Figs. 2.7 and 2.8, where N ranges from 1 to 1300, and $Z = 3.5$ s.

Using CMVA as the benchmark (as it is an exact solution), SMVA works better than Seidmann's approximation in all three cases. However, we also observe some errors for both of the approximate MVA algorithms from those figures, except for Figs. 2.6 and 2.8, where the largest errors are only -1.05% and -0.23%, respectively (negative means underestimate). The reason is that for those figures, the throughput is given by (2.5). As Z increases, the error in R has a smaller effect on the accuracy of X.

Fig. 2.3 Response time with $Z = 0$



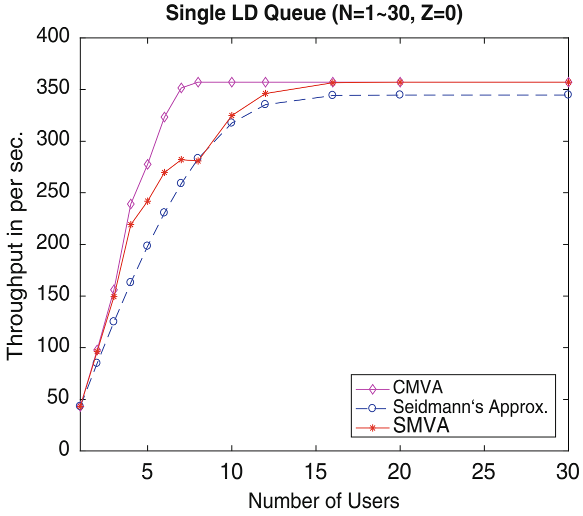


Fig. 2.4 Throughput with $Z = 0$

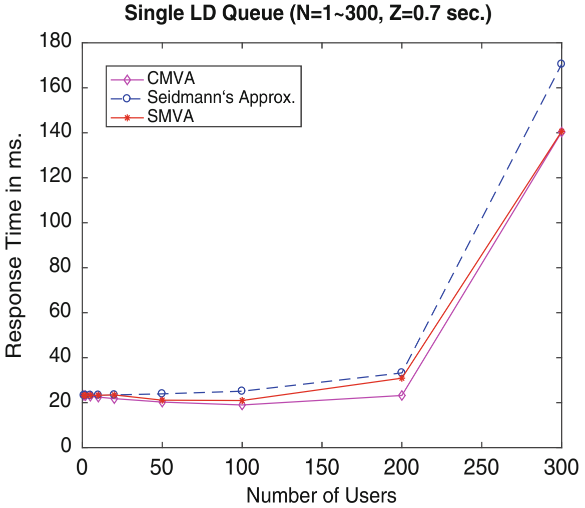


Fig. 2.5 Response time with $Z = 0.7\,\text{s}$

To verify the observations from Figs. 2.3, 2.4, 2.5, 2.6, 2.7 and 2.8, we calculate the Root-Mean-Square Percentage Error (RMSPE) for both SMVA and Seidmann's approximation in the three test sets. Here, RMSPE $= \sqrt{\sum_{i=1}^{T} E_i^2 / T}$, where E_i is the percentage error of the ith estimate, and T is the total number of estimates. The results can be found in Table 2.1, and they verify the two observations that we have in the figures:

- In terms of accuracy, SMVA works better than Seidmann's approximation in all cases.
- Errors in throughput are minor when Z is relatively larger than R.

Fig. 2.6 Throughput with $Z = 0.7\,\mathrm{s}$

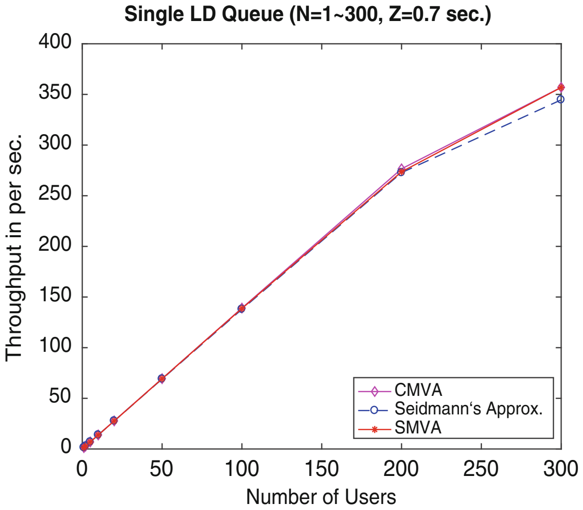
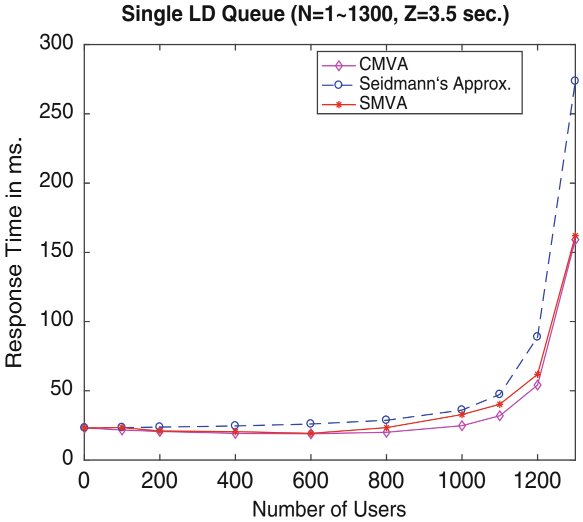


Fig. 2.7 Response time with $Z = 3.5\,\mathrm{s}$



We also would like to quantify some large errors from SMVA and Seidmann's approximation. In Figs. 2.3 and 2.4, the largest error for the mean response time of SMVA is 27.16%, and the largest error for the throughput of SMVA is −21.36% when $N = 8$. In contrast, the errors for both the mean response time and the throughput of Seidmann's approximation are the largest when $N = 4$ (46.78% and −31.87%, respectively). We have similar observations from Figs. 2.5, 2.6, 2.7 and 2.8. In Fig. 2.5, the largest error for the mean response time of SMVA is 33.17% when $N = 200$. In Fig. 2.7, the largest error for the mean response time of SMVA is 32.48% when $N = 1000$. Seidmann's approximation has its worst case when $N = 1300$, the error for the mean response time is 71.91%. As a conclusion,

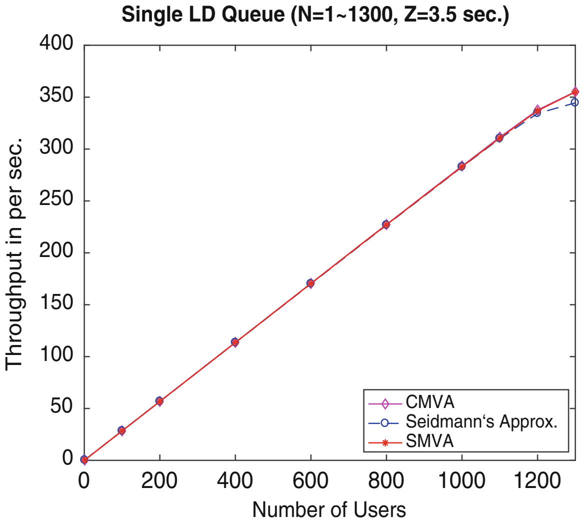


Fig. 2.8 Throughput with $Z = 3.5\,\mathrm{s}$

Table 2.1 RMSPEs in one LD queue

Test case		SMVA	Seidmann
$Z = 0.0\,\mathrm{s}$	R	12.93%	25.54%
	X	10.63%	18.72%
$Z = 0.7\,\mathrm{s}$	R	12.02%	20.47%
	X	0.38%	1.29%
$Z = 3.5\,\mathrm{s}$	R	15.24%	42.42%
	X	0.13%	1.03%

the SMVA algorithm works well when the system is under light or heavy loads. However, some errors are significant when the system is under intermediate loads.

2.5.2 *Two Load-Dependent Queues*

For the second queueing network, we add one more FES to the previous network. We derive the service demands of the second FES from the shopping web interaction workloads of TPC-W (as shown in Fig. 2.2). We choose the shopping workloads, because the service demand curve is close to (but not the same as) the one derived from the browsing workloads in the first FES (in Fig. 2.1), so that no single queue can dominate the performance in the network.

As discussed in Sect. 2.5.1, we vary the number of users and the mean think time in the system, and compare the mean response time and the throughput among the three candidate MVA algorithms. Since the results of throughputs are almost identical when Z is larger than zero (similar to Figs. 2.6 and 2.8), those results are not shown. The results of the first set are presented in Figs. 2.9 and 2.10, where N

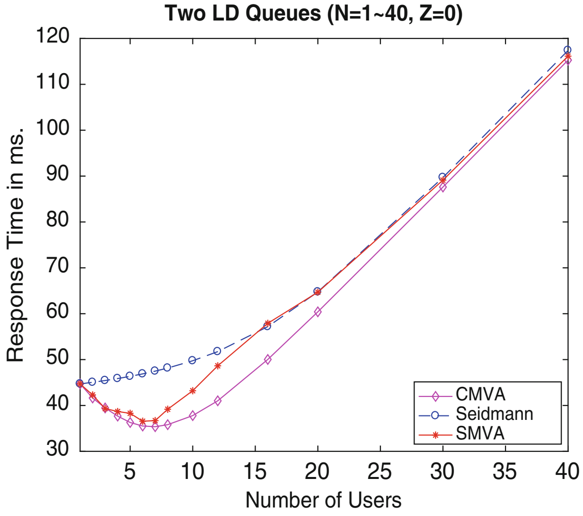


Fig. 2.9 Response time with $Z = 0$

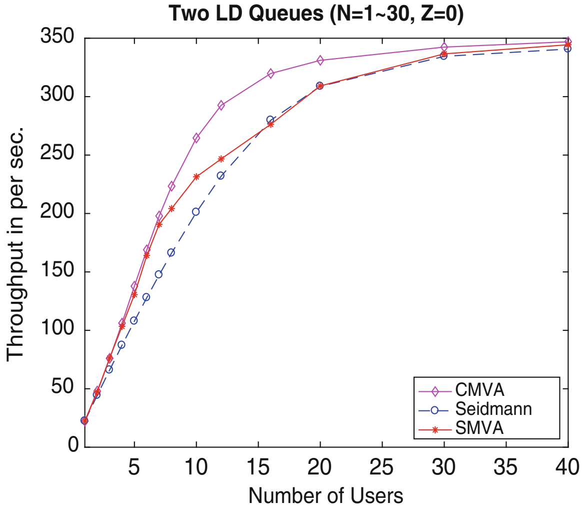


Fig. 2.10 Throughput with $Z = 0$

ranges from 1 to 40, and $Z = 0$. The results of the second set are presented in Fig. 2.11, where N ranges from 1 to 400, and $Z = 0.7$ s. The results of the third set are presented in Fig. 2.12, where N ranges from 1 to 1200, and $Z = 3.5$ s. Unlike the test set of $Z = 3.5$ s in Sect. 2.5.1, where we could have maximum 1300 jobs in the system, we cannot have 1300 jobs in this test case, because the calculation space requirement of the CMVA algorithm grows exponentially as the number of load-dependent queues increases. In this case, it is $O(MN^3)$, and it exceeds the maximum capacity of the memory on our machine. For instance, the initialization of the service demands for a single queue requires $N^3 \times 8$ bytes = 16.37 GB.

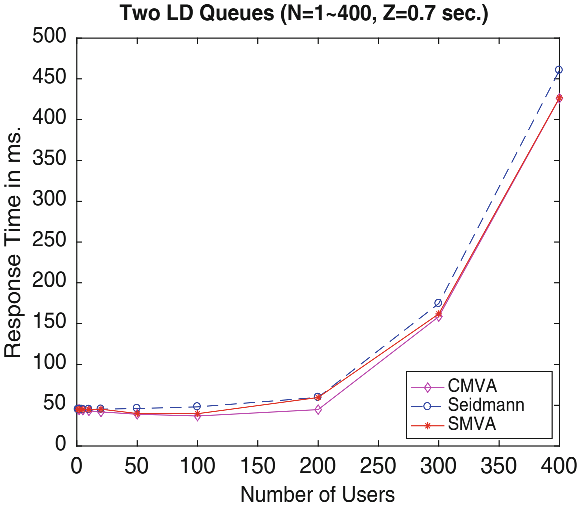


Fig. 2.11 Response time with $Z = 0.7\,\mathrm{s}$

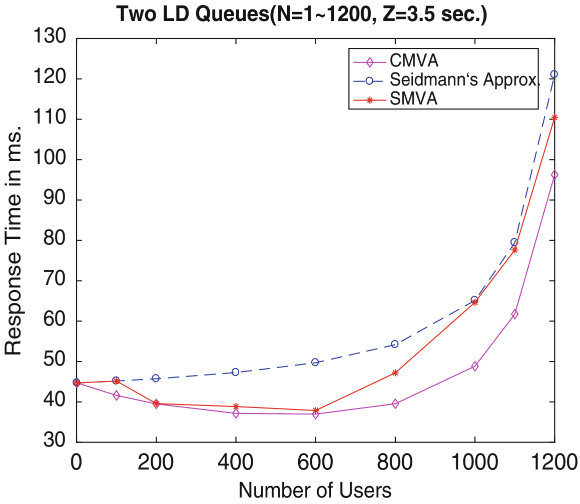


Fig. 2.12 Response time with $Z = 3.5\,\mathrm{s}$

As can be seen from Figs. 2.9, 2.10, 2.11 and 2.12, the results are quite consistent with those from Figs. 2.3, 2.4, 2.5, 2.6, 2.7 and 2.8, respectively. Similarly, we have three observations as follows:

- Compared to CMVA, SMVA works better than Seidmann's approximation in all three cases.
- As the value of the mean think time grows, the errors in estimated throughputs from SMVA decrease.
- Compared to the results under light and heavy workloads, larger errors are observed for SMVA under intermediate workloads.

Table 2.2 RMSPEs in two LD queues

Test case		SMVA	Seidmann
$Z = 0.0$ s	R	8.47%	22.15%
	X	7.40%	17.18%
$Z = 0.7$ s	R	11.23%	16.21%
	X	0.66%	1.39%
$Z = 3.5$ s	R	16.34%	26.27%
	X	0.26%	0.39%

Table 2.3 Largest error comparison for SMVA

Test case		Single LD queue	Two LD queues
$Z = 0.0$ s	R	27.16%	18.60%
	X	−21.36%	−15.68%
$Z = 0.7$ s	R	33.17%	33.33%
	X	−1.05%	−1.96%
$Z = 3.5$ s	R	32.48%	32.40%
	X	−0.23%	−0.44%

In Table 2.2, we show the RMSPEs for both SMVA and Seidmann's approximation in the three test sets. The results in Table 2.2 can be seen to verify the first two observations above. Compared to the results in Table 2.1, the SMVA algorithm performs better when $Z = 0$ in the network with two LD queues than in the network with a single LD queue in terms of the RMSPE.

In Table 2.3, we compare the largest errors of the SMVA algorithm in these two case studies. As can be seen, the SMVA algorithm performs better when Z is zero in the second case study, but has very similar results when Z is larger than zero. This observation is consistent with the observation from the RMSPEs in Tables 2.1 and 2.2. The underlying reason is not clear and is worth future study.

2.6 Multi-class Extension

For completeness, we extend the SMVA algorithm to the case of multi-class closed networks. As in single-class networks, the scheduling discipline in multi-class networks is also constrained to preserve the product-form nature of the steady-state distribution, for example the scheduling disciplines considered in the BCMP theorem [2] may be employed. We consider that there are C classes of transactions, where the job population vector is given by $\mathbf{n} = (n_1, \ldots, n_c, \ldots, n_C)$, where $0 \leq n_c \leq N_c$ and $1 \leq c \leq C$. The service demand of class c at the mth load-independent queue is given by $D^q_{m,c} = D_{m,c}(\bar{N}_{m,c})$. The service demand at the delay centre becomes

$$D^d_{m,c}(n_c) = \begin{cases} n_c D_{m,c}(n_c) - D_{m,c}(\bar{N}_{m,c}), & \text{if } n_c < \bar{N}_{m,c}, \\ (\bar{N}_{m,c} - 1) D_{m,c}(\bar{N}_{m,c}), & \text{if } n_c \geq \bar{N}_{m,c}. \end{cases}$$

Then, the multi-class SMVA iterates over all feasible $\mathbf{n} = (n_1, \ldots, n_C)$ such that $\sum_{c=1}^{C} n_c = n$ and $1 \leq n \leq N$ to compute the mean response times at load-independent queues:

$$R^q_{m,c}(\mathbf{n}) = D^q_{m,c}[1 + Q_m(\mathbf{n} - 1_c)].$$

Here, $\mathbf{n} - 1_c = (n_1, \ldots, n_c - 1, \ldots, n_C)$ is the job population vector with one less class c job in the system. The mean response time at a pure delay centre is $R^d_{m,c}(\mathbf{n}) = D^d_{m,c}(\lceil Q^e_m(\mathbf{n}) \rceil)$, where

$$Q^e_m(\mathbf{n}) = \begin{cases} 1, & \text{if } n_c = 1, \\ \dfrac{n_c}{n_c - 1} Q^o_m(\mathbf{n} - 1_c), & \text{if } n_c > 1. \end{cases}$$

The system throughput of class c is calculated by

$$X_c(\mathbf{n}) = n_c / \left\{ Z_c + \sum_{m=1}^{M} [R^q_{m,c}(\mathbf{n}) + R^d_{m,c}(\mathbf{n})] \right\}.$$

The mean queue length at the mth load-independent queue is

$$Q^a_m(\mathbf{n}) = \sum_{c=1}^{C} X_c(\mathbf{n}) \cdot R^q_{m,c}(\mathbf{n}).$$

Finally, the mean queue length at the original load-dependent queue is

$$Q^o_m(\mathbf{n}) = \sum_{c=1}^{C} X_c(\mathbf{n}) \cdot [R^q_{m,c}(\mathbf{n}) + R^d_{m,c}(\mathbf{n})].$$

Both the time and space complexities of the multi-class SMVA algorithm are $O(MN^C)$. Due to the complexities, we have not evaluated the accuracy of the multi-class model in a case study.

2.7 Conclusions

In this paper, we present the SMVA algorithm in both a single-class model and a multi-class model. Compared to the CMVA algorithm and Seidmann's approximation, the SMVA algorithm has two advantages:

- The time and space complexities of SMVA are a significant improvement over CMVA, especially when the number of jobs in the system is very large, or when the number of load-dependent queues is larger than one.
- The SMVA algorithm is better able to handle cases when the service demands of a load-dependent node do not have a linear relationship.

In terms of accuracy, we also have two additional observations about the SMVA algorithm:

- The SMVA algorithm works as well as the CMVA algorithm when the system is under light or heavy loads. However, the errors of the SMVA algorithm increase when the system is under intermediate loads (but it still performs better than Seidmann's approximation).
- When the mean think time increases, the SMVA algorithm might produce less accurate estimates of the mean response times under intermediate load. In contrast, the estimated throughput becomes more accurate.

The accuracy of SMVA under intermediate loads is closely linked to the accuracy of the underlying approximations. It is inspired by Seidmann's approximation. Consequently, it behaves as Seidmann's approximation under intermediate workloads. In addition, it employs the assumption in the Bard-Schweitzer approximation to estimate the mean number of jobs at delay centres, which may also add errors to the results.

Acknowledgements The work reported here was supported by the Natural Sciences and Engineering Research Council of Canada.

Appendix

In Algorithm 1, we estimate the mean number of jobs at a delay centre, $Q^e_m(n)$. In the same iteration, new values are calculated as $Q^o_m(n)$. A natural thought would be to add a comparison between these two values, similar to a technique in the Bard-Schweitzer approximation. To accomplish this, we propose an alternative SMVA algorithm (A-SMVA) for a single-class system. The details of A-SMVA can be seen in Algorithm 2. Compared to SMVA, A-SMVA differs as follows:

- Iterations over $n = 1 \rightarrow N$ are removed. Instead, we focus only on the performance metrics with N jobs in the system.
- Initialize $Q^e_m(N)$ with estimated values, e.g., $N/(M+1)$.
- Employ (2.4) to replace $Q^a_m(N-1)$ by $Q^q_m(N)$ in (2.3), which is $Q^q_m(N) \times (N-1)/N$.
- Choose an error criterion—ϵ, e.g., 0.01.
- Compare the difference between $Q^e_m(N)$ and $Q^o_m(N)$, and compare the difference between $Q^q_m(N)$ and $Q^a_m(N)$. If the maximum difference is larger than ϵ, replace $Q^e_m(N)$ by $Q^o_m(N)$, and $Q^q_m(N)$ by $Q^a_m(N)$. Otherwise, stop the iteration.

Algorithm 2 The single-class A-SMVA algorithm

```
Input:
$Z, M, N, D_m(n), \bar{N}_m, \epsilon$
Output:
$Q_m^o(N), X(N), R(N)$
Condition:
$N \cdot D_m(N) \geq D_m(\bar{N}_m), \forall m$
Initialization:
$Q_m^a(N) = 0, \forall m = 1, \ldots, M$
$Q_m^o(N) = N/(M+1), \forall m = 1, \ldots, M$
Iteration:
  for $m = 1 \to M$ do
    $D_m^q = D_m(\bar{N}_m)$
    $D_m^d(N) = \begin{cases} N \cdot D_m(N) - D_m(\bar{N}_m), & \text{if } N < \bar{N}_m \\ (\bar{N}_m - 1)D_m(\bar{N}_m), & \text{if } N \geq \bar{N}_m \end{cases}$
  end for
  while $\max_i\{|Q_m^e(N) - Q_m^o(N)|\} > \epsilon$ or $\max_i\{|Q_m^q(N) - Q_m^a(N)|\} > \epsilon$ do
    for $m = 1 \to M$ do
      $Q_m^e(N) = Q_m^o(N)$
      $Q_m^q(N) = Q_m^a(N)$
      $R_m^q(N) = D_m^q[1 + \frac{N-1}{N} Q_m^q(N)]$
      $R_m^d(N) = D_m^d(\lceil Q_m^e(N) \rceil)$
    end for
    $X(N) = N/\{Z + \sum_{m=1}^{M}[R_m^q(N) + R_m^d(N)]\}$
    for $m = 1 \to M$ do
      $Q_m^a(N) = X(N) \cdot R_m^q(N)$
      $Q_m^o(N) = X(N) \cdot [R_m^q(N) + R_m^d(N)]$
    end for
  end while
  $R(N) = \sum_{m=1}^{M}[R_m^q(N) + R_m^d(N)]$
```

We set $\epsilon = 0.01$, and compare the results of A-SMVA with the results of SMVA with the same input parameters in Sect. 2.5.1. The estimated mean response times from A-SMVA are slightly larger than the results from SMVA, but they have very similar trends.

When N is very large, A-SMVA can be efficient, because it avoids the iteration over $n = 1 \to N$. However, we have two concerns about A-SMVA:

- The initial values of $Q_m^e(N)$ may significantly affect the outputs. For example, if they are too close to zero, the whole iteration will be skipped.
- The chosen value of ϵ may have a significant effect on the outputs. If ϵ is too big, we have less iterations, but sacrifice accuracy. If ϵ is too small, it may not converge in some instances (although we have not observed this).

As a summary, we provide one more numerically stable approach to determine the performance metrics for closed queueing networks with load-dependent queues. One can adopt SMVA or A-SMVA depending on the requirements.

References

1. Y. Bard, Some extensions to multiclass queueing network analysis, in *Proceedings of the 3rd International Symposium on Modelling and Performance Evaluation of Computer Systems: Performance of Computer Systems* (North-Holland Publishing Co., New York, 1979), pp. 51–62
2. F. Baskett, K.M. Chandy, R.R. Muntz, F.G. Palacios, Open, closed, and mixed networks of queues with different classes of customers. J. ACM **22**(2), 248–260 (1975)
3. G. Casale, A note on stable flow-equivalent aggregation in closed networks. Queueing Syst. **60**(3–4), 193–202 (2008)
4. G. Casale, G. Serazzi, Stabilization techniques for load-dependent queueing networks algorithms, in *Communication Networks and Computer Systems: A Tribute to Professor Erol Gelenbe*, Chap 8, ed. by J.A. Barria (Imperial College Press, London, 2006), pp. 127–141
5. G. Casale , J.F. Pérez, W. Wang, QD-AMVA: evaluating systems with queue-dependent service requirements. Perform. Eval. **91**, 80–98 (2015)
6. K.M. Chandy, C.H. Sauer, Computational algorithms for product form queueing networks. Commun. ACM **23**(10), 573–583 (1980)
7. K.M. Chandy, U. Herzog, L. Woo, Parametric analysis of queuing networks. IBM J. Res. Dev. **19**(1), 36–42 (1975)
8. Y. Chen, S. Iyer, X. Liu, D. Milojicic, A. Sahai, SLA decomposition: translating service level objectives to system level thresholds, in *Proceedings of the 4th International Conference on Autonomic Computing* (IEEE, 2007), pp. 3–13
9. Y. Chen , S. Iyer , X. Liu , D. Milojicic , A. Sahai, Translating service level objectives to lower level policies for multi-tier services. Clust. Comput. **11**(3), 299–311 (2008)
10. T. Horvath, TPC-W J2EE implementation (2008). http://www.cs.virginia.edu/~th8k. Last accessed 27 Sept 2017
11. K. Hoyme, S.C. Bruell, P. Afshari, R.Y. Kain, A tree-structured mean value analysis algorithm. ACM Trans. Comput. Syst. **4**(2), 178–185 (1986)
12. S.S. Lavenberg, M. Reiser, Stationary state probabilities at arrival instants for closed queueing networks with multiple types of customers. J. Appl. Probab. **17**(4), 1048–1061 (1980)
13. X. Liu, J. Heo, L. Sha, Modeling 3-tiered web applications, in *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005* (IEEE, 2005), pp. 307–310
14. D.A. Menascé, V.A. Almeida, L.W. Dowdy, L. Dowdy, *Performance by Design: Computer Capacity Planning by Example* (Prentice Hall PTR, Upper Saddle River, 2004)
15. M. Reiser, Mean-value analysis and convolution method for queue-dependent servers in closed queueing networks. Perform. Eval. **1**(1), 7–18 (1981)
16. M. Reiser, S.S. Lavenberg, Mean-value analysis of closed multichain queuing networks. J. ACM **27**(2), 313–322 (1980)
17. P. Schweitzer, Approximate analysis of multiclass closed networks of queues, in *Proceedings of International Conference on Stochastic Control and Optimization* (Free University, Amsterdam, 1979), pp. 25–29
18. A. Seidmann, J. Paul, S. Shalev-Oren, Computerized closed queueing network models of flexible manufacturing systems: a comparative evaluation. Large Scale Syst. **12**, 91–107 (1987)
19. K.C. Sevcik, I. Mitrani, The distribution of queuing network states at input and output instants. J. ACM **28**(2), 358–371 (1981)
20. S. Tucci, C.H. Sauer, The tree MVA algorithm. Perform. Eval. **5**(3), 187–196 (1985)
21. L. Zhang, D.G. Down, A stable mean value analysis algorithm for closed systems with load-dependent queues, in *Proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools* (ACM, New York, 2016), pp. 178–181