Production, Manufacturing, Transportation and Logistics

Stochastic modeling of parallel process flows in intra-logistics systems: Applications in container terminals and compact storage systems

Govind Lal Kumawat\textsuperscript{a,}\textsuperscript{*}, Debjit Roy\textsuperscript{b,c}, René De Koster\textsuperscript{c}, Ivo Adan\textsuperscript{d}

\textsuperscript{a} Indian Institute of Management Udaipur, Rajasthan 313001, India
\textsuperscript{b} Indian Institute of Management Ahmedabad, Gujarat 380015, India
\textsuperscript{c} Rotterdam School of Management, Erasmus University, Rotterdam 3062 PA, the Netherlands
\textsuperscript{d} Industrial Engineering & Innovation Sciences, Eindhoven University of Technology, Eindhoven, the Netherlands

\section{A R T I C L E   I N F O}
Article history:
Received 10 July 2019
Accepted 3 August 2020
Available online 8 August 2020

Keywords:
Logistics
Queueing
Parallel processes
Container terminals
Compact storage

\section{A B S T R A C T}
Many intra-logistics systems, such as automated container terminals, distribution warehouses, and cross-docks, observe parallel process flows, which involve simultaneous (parallel) operations of independent resources while processing a job. When independent resources work simultaneously to process a common job, the effective service requirement of the job is difficult to estimate. For modeling simplicity, researchers tend to assume sequential operations of the resources. In this paper, we propose an efficient modeling approach for parallel process flows using two-phase servers. We develop a closed queuing network model to estimate system performance measures. Existing solution methods can evaluate the performance of closed queuing networks that consist of two-phase servers with exponential service times only. To solve closed queuing networks with general two-phase servers, we propose new solution methods: an approximate mean value analysis and a network aggregation/dis-aggregation approach. We derive insights on the accuracy of the solution methods from numerical experiments. Although both solution methods are quite accurate in estimating performance measures, the network aggregation dis-aggregation approach consistently performs best. We illustrate the proposed modeling approach for two intra-logistic systems: a container terminal with automated guided vehicles and a shuttle-based compact storage system. Results show that approximating the simultaneous operations as sequential operations underestimates the container terminal throughput on average by 28\% and at maximum up to 47\%. Similarly, considering sequential operations of the resources in the compact storage system results in an underestimation of the throughput capacity up to 9\%.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

Stochastic models are widely used to evaluate the performance of intra-logistics systems, such as automated container terminals (Gupta, Roy, De Koster, & Parhi, 2017; Hoshino, Ota, Shinozaki, & Hashimoto, 2007; Mishra, Roy, & Van Ommeren, 2017), distribution warehouses (Heragu, Cai, Krishnamurthy, & Malmborg, 2011; Lerher, Ekren, Dukic, & Rosi, 2015; Marchet, Melacini, Perotti, & Tappia, 2012), and cross-docks (Bartholdi III & Gue, 2000). In many intra-logistics systems, the processing of a job involves parallel process steps where different resources can work independently to process the job. This is the case when the processing of a job can be partly executed without the job being physically present (e.g., setup of a resource while the job is arriving at the resource). Upon completion of the simultaneous operations (parallel process flows), the resources may or may not need to synchronize for completing the remaining process steps. A hard-coupling or synchronization between resources is essential if there is no intermediate storage buffer, and one resource needs to hand over the job to the subsequent resource for further processing. On the other hand, synchronization of resources is not required if one resource hands over the unfinished job to a buffer from where the second resource can pick the job for further processing.

An example of simultaneous operations without an intermediate buffer can be found in automated container terminals with automated guided vehicles (AGVs) (Roy, Gupta, & De Koster, 2015). Here, unloading operations of a container from a vessel can be divided into two phases: (1) fetching the container by a quay crane and (2) dropping it on an AGV. The movement of an empty AGV from the stackside (or dwell point) to the quayside and the retrieval of the container by a quay crane can be executed simultaneously. Next, the AGV and the quay crane must synchronize to
load the container on the AGV. This phase requires hard-coupling of the AGV and the quay crane.

Similarly, in a shuttle-based compact storage system (Tappia, Roy, De Koster, & Melacini, 2017), a storage or retrieval transaction within a tier involves working of two independent resources: a shuttle and a transfer car. When a shuttle retrieves a load from its current storage lane, it picks-up the load and travels to the cross-aisle. Simultaneously, the transfer car travels in the cross-aisle to the storage lane from its dwell point. Once both arrive at the cross-aisle location, the transfer car picks-up the shuttle and carries it to the outbound buffer. This process requires hard-coupling of the shuttle and the transfer car.

In practice, simultaneous operations of independent resources while serving a job have performance benefits over their sequential operations (Hu et al., 2005). However, stochastic modeling of the simultaneous operations is quite difficult. Since different resources can execute part of a job simultaneously, the effective service requirement of the job is hard to estimate. To overcome this difficulty and to make the analysis simpler, existing studies typically use ‘sequential’ modeling approach, which assumes sequential operations of different resources that can work at least partly simultaneously. For example, Yang, Choi, and Ha (2004), Roy and De Koster (2014), Dhingra, Roy, and De Koster (2017), Roy et al. (2015), and Gupta et al. (2017) assume sequential operations of quay/stack cranes and AGVs while modeling the seaside operations at an automated container terminals with AGVs. They assume that the quay crane’s container retrieval process can only start after an empty AGV arrives at the quayside. Similarly, Tappia et al. (2017) assume sequential operations between the transfer car and the shuttles in their stochastic model of the compact storage system.

In this paper, we develop a ‘parallel’ modeling approach that is efficient in modeling simultaneous operations of independent resources. Specifically, our approach is applicable to model simultaneous operations where resources must be synchronized (hard-coupled) upon completion of their simultaneous operations. Such operations can be found in many environments, e.g., in intra-logistics systems. Simultaneous operations can be modeled using simulation. Developing simulation models is relatively easy, but time consuming. In addition, they are computationally expensive when the system performance is obtained with a large number of scenarios. We propose an analytical closed queuing network with two-phase servers to realistically model the simultaneous operations of resources and to estimate the performance measures of the system. To analyze the resulting queuing network, we develop two solution methods: an approximate mean value analysis and a network aggregation dis-aggregation based approach. We validate the solution methods using numerical experiments and derive several insights on their accuracy. Results indicate that the solution methods are quite accurate in estimating the performance measures, but the network aggregation dis-aggregation based approach is the better one.

We demonstrate the advantages of the parallel modeling approach over the sequential modeling approach using two applications: an automated container terminal with AGVs (Fig. 1(a)) and a shuttle-based compact storage system (described in Section 5). Fig. 1(b) and (c) shows the sequential modeling approach and the parallel modeling approach for the unloading operations at an automated container terminal with AGVs, respectively. We model the simultaneous operations of quay/stack cranes and AGVs and estimate the performance measures. Similarly, in the compact storage system, we model the simultaneous operations of shuttles and the transfer car in the storage tier and estimate the performance measures (see Section 5).

The rest of the paper is organized as follows. In Section 2, we present the literature review. The modeling approach and solution methods are described in Section 3. In Sections 4 and 5, we develop stochastic models for the seaside operations at an automated container terminal with AGVs and for a shuttle-based compact storage system, respectively. We conclude the paper in Section 6.

2. Literature review

In reviewing the stochastic modeling literature, we primarily focus on the systems which observe simultaneous operations of different resources. Specifically, studies on intra-logistics systems are more relevant to this paper. Developing stochastic models for intra-logistics systems is a well-established area of research. Vis and De Koster (2003) present a comprehensive literature review of container terminal operations covering a variety of decision problems related to resource planning, routing, performance, and scheduling. More reviews of container terminals can be found in Steenken, Voß, and Stahlbock (2004) and Carlo, Vis, and Roodbergen (2014). Yang et al. (2004) develop simulation models for comparing the performance of AGVs and automated lift vehicles (ALVs). They assume sequential movements of AGVs/ALVs and quay/stack cranes. Petering and Murty (2009) and Petering (2009) use discrete event simulation models to evaluate the effect of the block stack layout and the yard crane deployment on the performance of a container terminal. Kemmke (2012) and Taner, Kulak, and Koyuncuğlu (2014) evaluate strategic decisions related to the block stack layout, resource allocation, and dispatching rules of transporters using simulation models. Few researchers have developed analytical models for evaluating container terminal operations. Roy et al. (2015) develop a non-linear flow-based analytical queuing model to evaluate the performance of a container terminal with AGVs. In another study, Roy and De Koster (2014) develop semi-open queuing network (SOQN) models for a container terminal with AGVs and ALVs. Dhingra et al. (2017) study the effect of cooperation between quay cranes on the container terminal performance using a closed queuing network. These studies assume sequential operations of AGVs/ALVs and quay/stack cranes. Mishra et al. (2017) propose an inter-terminal transport model for container transportation between terminals. While developing an SOQN network, they assume a sequential flow of unloading containers and trucks movement from the central dwell point to the origin station.

Automated vehicle-based storage and retrieval systems (AVS/RSs) are also intra-logistics systems, which have received considerable attention from researchers. A literature review on models used for the performance evaluation of AVS/RSs is published by Gagliardi, Renaud, and Ruiz (2012). Heragu, Cai, Krishnamurthy, and Malmborg (2009), Malmborg (2002), Marchet et al. (2012), and Lerher et al. (2015) develop stochastic models to evaluate performance of AVS/RSs. These studies assume sequential working of the lift and the tier-vehicles. In a storage transaction, the lift first transports a load to the inbound buffer at the designated tier and then requests the vehicle. A retrieval transaction first requests to a vehicle and then the lift. Some studies develop analytical models, whereas some studies use simulation models. Ekren, Heragu, Krishnamurthy, and Malmborg (2014), Roy, Krishnamurthy, Heragu, and Malmborg (2012), and Cai, Heragu, and Liu (2014) develop semi-open queuing networks (SOQNs) to evaluate performance of AVS/RSs. To solve SOQNs, Ekren et al. (2014) and Cai et al. (2014) use matrix-geometric method, whereas Roy et al. (2012) develop a network decomposition based approach to solve SOQNs. Ekren and Heragu (2012) develop simulation models to compare the performance of AVS/RSs and crane-based storage and retrieval systems (CBAS/RSs).

Simultaneous operations considered in this paper are closely aligned with Hu et al. (2005). They evaluate the performance of a split-platform-AVS/RS, in which a storage or retrieval transaction is performed by the simultaneous movement of a vertical platform.
(VP) and a horizontal platform (HP). Due to no intermediate buffer, synchronization between the VP and the HP is essential to transfer the load. While Hu et al. (2005) develop deterministic models for analyzing simultaneous operations, we develop stochastic models that are close to reality and take into account the interactions among resources. Zou, Xu, Cong, and De Koster (2016) develop a fork-join based queuing model to evaluate the performance of AVS/RSs, in which the lift and the tier-vehicles can operate simultaneously. AVS/RSs have intermediate buffer spaces for temporary storage of loads, therefore loads can be transferred between the lift and tier-vehicles without synchronization. However, our model is useful for different systems where two resources can operate simultaneously and no intermediate buffer is present, hence resources must be hard-coupled (synchronized) to hand over the unfinished job. Simultaneous matching of customers (or orders) with resources has been modeled in the literature using fork-join synchronization queues. For example, Roy (2016) describes such applications in manufacturing, health care (patient matching with an transport vehicle), restaurant (order matching with a chef) etc. However, in these applications, resources work in parallel, but do not involve hard-coupling to hand-over unfinished jobs.

Our literature review shows that most studies use a sequential modeling approach for analyzing the performance of intra-logistics systems wherein different resources work simultaneously to process a job. Differing from the existing literature, we develop a parallel modeling approach for the performance evaluation of intra-logistics systems, taking into account the impact of simultaneous operations and hard-coupling of resources. To model simultaneous operations, we use a special type of two-phase server introduced in Van Doremalen (1986). The author discusses approximate solution methods for queuing network models with exponential service times. Our solution method, the approximate mean value analysis, builds on the mean value approach proposed by Van Doremalen (1986). However, we derive a closed form expression for the network throughput and propose an algorithm which requires fewer iterations. We also develop an algorithm for closed queuing networks with general service times. We can summarize our contributions as follows. In the online supplement, we also contrast our work with the literature.

1. Modeling approach for parallel process flows: We develop an approach to model the simultaneous operations of resources in intra-logistics systems using queuing based analytical models.

2. Solution methods for closed queuing networks with two-phase servers: We develop two solution methods: an approximate mean value analysis and a network aggregation dis-aggregation approach. We derive several insights on the accuracy of the solution methods from numerical experiments.

3. Applications of the parallel modeling approach: We apply our modeling approach on two widely studied intra-logistics systems: an automated container terminal with AGVs and a shuttle-based compact storage system. In literature, these systems are modelled using the sequential modeling approach. We use the parallel modeling approach and show that it leads to more accurate estimates than the predominately used sequential modeling approach.

3. Modeling approach and solution methods

In this section, we develop an analytical two-phase server based closed queuing network for modeling the parallel process flows in intra-logistics systems. We propose two solution methods to analyze the resulting queuing network and compare their accuracy based on the numerical results.

3.1. Modeling approach for parallel process flows

We explain our modeling approach using a simple sub-system that consists of two-tandem work stations. In Fig. 2, we show the parallel and sequential operations in the work stations and corresponding analytical queuing models. Station 1 operates in one stage (no preparation is required to serve the job), whereas Station 2 operates in two independent stages: preparation (setup) and synchronization (processing). The preparation stage can be executed while the job is on the way to arrive at Station 2. As a result, the effective service requirement for the job is reduced. If a job arrives before the station finishes the preparation task, then the job has to wait and vice-versa. Essentially, the job arrival and the station preparation task are two simultaneous (parallel) operations. Once both these operations are finished, the job synchronizes with the station resources and completes service. Every job requires an independent preparation before executing its synchronization stage. Since Station 1 works in one stage, it can be modeled as a one-phase server. In a sequential modeling approach, Station 2 is not permitted to execute the preparation stage until a job arrives at the station. With such an assumption, the service requirement of
the job is equal to the sum of execution times of the independent stages (preparation and synchronization). Therefore, Station 2 can also be modeled as a one-phase server. However, due to the simultaneous preparation and job arrival in the real system, the service requirement at Station 2 varies between the synchronization time and the sum of the preparation and synchronization times. To model such operations, we use a two-phase server that has two working phases: setup phase and processing phase. The setup phase can be executed in the absence of the job. The processing phase is executed when a job is present with its setup phase finished. Thus, we can represent the preparation and synchronization stage of the Station 2 by the setup phase and the processing phase of a two-phase server.

3.2. Solution methods for a closed queuing network with two-phase servers

In this section, we develop two solution methods for analyzing a closed queuing network with two-phase servers. We assume that each service station has one server of either two-phase type, one-phase type, or load-dependent type. The first method (Section 3.2.1) is an approximate mean value analysis method, and the second method (Section 3.2.2) is a network aggregation dis-aggregation approach, which is based on flow-equivalence. Notations used in this section are described in Table 1.

3.2.1. Approximate mean value analysis (AMVA)

We first consider exponential service times for all servers, and in the subsequent section, we consider general service times.

(a) Exponential service times: Consider an arbitrary closed queuing network with \( m_d \) one-phase servers (OPSS), \( m_d \) load-dependent exponential servers (LDSs), and \( m_t \) two-phase servers (TPSs). The network contains \( N \) jobs which receive service at different servers in the network and circulate in a closed loop. A job visits to the \( i \)th OPS, the \( j \)th LDS, and the \( k \)th TPS with visit ratios \( \epsilon_{ij} \), \( \epsilon_{ij} \), and \( \epsilon_{jk} \), respectively, where \( i = 1, 2, \ldots, m_d \), \( j = 1, \ldots, m_t \), and \( k = 1, \ldots, m_t \). We represent the throughput of the closed network with \( n \) jobs by \( \lambda_{in} \), where \( n = 1, \ldots, N \). We obtain the mean sojourn time expression of a job at an OPS, an LDS, and a TPS as follows.

Mean sojourn time of a job at an OPS: Let the mean service time at the \( i \)th OPS be given by \( w_{i-1} \). Let an arriving job to the \( i \)th OPS finds \( Q_{in} \) jobs (on average) at the server including the job in service when the network has \( n \) jobs. We denote the mean sojourn time at the \( i \)th OPS with \( n \) jobs in the network by \( S_{in} \). Using the mean value approach, we can write \( S_{in} \) as follows

\[
S_{in} = Q_{in-1} W_{i-1}^{-1} + W_{i-1}^{-1}. \tag{1}
\]

Mean sojourn time of a job at an LDS: Let \( w_r \) be the mean service time of the \( j \)th LDS with \( r \) jobs at the server, \( r = 1, \ldots, n \). From Lazowska, Zahorjan, Graham, and Sevcik (1984), the mean sojourn time at the \( j \)th LDS with \( n \) jobs in the network, \( S_{jn} \), can be written as:
as

\[ S^r_{k,n} = \sum_{r=1}^{n} ru_j^{-1}(r)\delta_j(r-1|n-1) \]  

(2)

where \( \delta_j(r|n) \) is the probability that \( r \) jobs are present at the \( j \)th LPS when there are \( n \) jobs in the network. The expression of \( \delta_j(r|n) \) is given as follows

\[ \delta_j(r|n) = \begin{cases} e^{\alpha_j}u_j^{-1}(r)\delta_j(r-1|n-1) & \text{for } r = 1, \ldots, n \\ 1 - \sum_{\zeta=1}^{\infty} \delta_j(\zeta|n) & \text{for } r = 0. \end{cases} \]

(3)

Mean sojourn time of a job at a TPS: Let \( \mu_k^{-1} \) and \( v_k^{-1} \) be the mean setup time and the mean processing time of the kth TPS, respectively. Let an arriving job to the kth TPS finds on average \( Q^r_{k,n} \) jobs at the server (including the job in the processing phase) with \( n \) jobs in the network. When a job arrives at a TPS, it finds the server in either setup, processing, or idle state. Consider that an arriving job sees the kth TPS in the setup phase with probability \( p_{k,n-1} \) and in the processing phase with probability \( q_{k,n-1} \) with \( n \) jobs in the network. The server can be idle with probability \( 1 - p_{k,n-1} - q_{k,n-1} \). By the mean value approach, the mean sojourn time at the kth TPS, \( S^r_{k,n} \), can be given as follows

\[ S^r_{k,n} = (Q^r_{k,n-1} - q_{k,n-1})(\mu_k^{-1} + v_k^{-1}) + q_{k,n-1}(v_k^{-1} + \mu_k^{-1}) + p_{k,n-1}\mu_k^{-1} + v_k^{-1}. \]

(4)

In Eq. (4), the first term, \( (Q^r_{k,n-1} - q_{k,n-1})(\mu_k^{-1} + v_k^{-1}) \), denotes the mean service (setup and processing) time of all jobs (excluding the job in processing phase) at the kth TPS seen by an arriving job. The second term, \( q_{k,n-1}(v_k^{-1} + \mu_k^{-1}) \), denotes the mean residual time (mean waiting time before the next job enters in the processing phase) if the server is in the processing phase at the job arrival instance. The third term, \( p_{k,n-1}\mu_k^{-1} \), denotes the mean residual time if the server is in the setup phase at the job arrival instance. The fourth term, \( v_k^{-1} \), denotes the mean processing time of the arriving job. After simplifying Eq. (4), we get

\[ S^r_{k,n} = Q^r_{k,n-1}(\mu_k^{-1} + v_k^{-1}) + p_{k,n-1}\mu_k^{-1} + v_k^{-1}. \]

(5)

The probability that an arriving job finds the server in the setup phase is equal to the utilization of the setup phase. Then, \( p_{k,n-1} = X_{n-1}\mu_k^{-1}u_k^{-1} \) and

\[ S^r_{k,n} = Q^r_{k,n-1}(\mu_k^{-1} + v_k^{-1}) + X_{n-1}(\mu_k^{-1})^2u_k^{-1} + v_k^{-1}. \]

(6)

Notice that when \( n = 1 \), \( Q^r_{k,n-1} = 0 \), \( X_{n-1} = 0 \) and \( S^r_{k,n-1} = v_k^{-1} \). This implies that \( p_{k,n-1} = 0 \) for \( \mu_k^{-1} > 0 \) (the arrival job never finds the server in the setup phase). This is only possible when the job interarrival time at kth TPS is always more than the setup time. Due to variability in the setup time and the job interarrival time, the latter may not hold true. To overcome this difficulty, we approximate \( X_{n-1} \) by \( X_n \), which seems a reasonable choice for any closed queuing network, because as \( n \) increases, \( X_{n} - X_{n-1} \) approaches zero. The mean sojourn time at the kth TPS, \( S^r_{k,n} \), is given as follows

\[ S^r_{k,n} = Q^r_{k,n-1}(\mu_k^{-1} + v_k^{-1}) + X_n(\mu_k^{-1})^2u_k^{-1} + v_k^{-1}. \]

(7)

Performance measures: From the mean sojourn time expressions, we obtain the expression of \( X_n \) as follows

\[ X_n = \sum_{j=1}^{m} S^0_{j,n}e_j^0 + \sum_{j=1}^{m} S^r_{j,n-1}e_j^0 + \sum_{j=1}^{m} S^r_{j,n}e_j^1. \]

(8)

The denominator term in Eq. (8) represents the mean cycle time of a job in the closed queuing network with \( n \) jobs. After substituting \( S^0_{j,n} \), \( S^r_{j,n} \) and \( S^r_{j,k,n} \) in Eq. (8), we get the following quadratic equation

\[ X_n = \frac{n}{\alpha_n + X_n \sum_{k=1}^{m} (\mu_k^{-1}e_k^0)^2}. \]

(9)

where \( \alpha_n = \sum_{k=1}^{m} (Q^r_{k,n-1}(\mu_k^{-1} + v_k^{-1}) + v_k^{-1})e_k^0 + \sum_{j=1}^{m} S^r_{j,n}e_j^0 + \sum_{j=1}^{m} S^r_{j,n}e_j^1 \). Since \( X_n \to 0 \), we consider the non-negative root of the quadratic equation. This results in the following closed form expression of \( X_n \).

\[ X_n = \frac{1}{2 \sum_{k=1}^{m} (\mu_k^{-1}e_k^0)^2} \left( \sqrt{\alpha_n^2 + 4n \sum_{k=1}^{m} (\mu_k^{-1}e_k^0)^2} - \alpha_n \right). \]

(10)

**Proposition 1.** If the mean of each two-phase server in the closed queuing network approaches zero, the two-phase server is equivalent to a one-phase server. Therefore, if \( \mu_k^{-1} \to 0 \) for all \( k = 1, \ldots, m \), the throughput given in Eq. (10) reduces to the one obtained in standard MVA and is given as

\[ \lim_{\mu_k^{-1} \to 0} X_n = \sum_{k=1}^{m} (Q^r_{k,n-1}v_k^{-1} + v_k^{-1})e_k^0 + \sum_{j=1}^{m} S^r_{j,n}e_j^0 + \sum_{j=1}^{m} S^r_{j,n}e_j^1. \]

(11)

The proof is provided in the online supplement. A TPS with no setup time is equivalent to an OPS. In this case, Proposition 1 implies that the expression of \( X_n \) in Eq. (10) is equal to the one obtained in the standard MVA and is given by Eq. (11), where the denominator, \( \sum_{k=1}^{m} (Q^r_{k,n-1}v_k^{-1} + v_k^{-1})e_k^0 + \sum_{j=1}^{m} S^r_{j,n}e_j^0 + \sum_{j=1}^{m} S^r_{j,n}e_j^1 \) is the sum of mean sojourn times at all servers (or mean cycle time of a job) in the network. We now obtain the expressions of the mean queue length and the utilization of each server. From Little’s law, we get the mean queue length at the ith TPS as \( Q^o_{i,n} = X_nS^o_{i,n}e_i^0 \) and the utilization \( U^o_{i,n} = X_nw_i^{-1}e_i^0 \). Similarly, we obtain the mean queue length at the jth LPS as \( Q^r_{j,n} = X_nS^r_{j,n}e_j^0 \) and the utilization \( U^r_{j,n} = 1 - \delta_j(0|n) \). For a TPS, we calculate \( Q^o_{i,n} = X_nS^o_{i,n}e_i^0 \). However, finding the expression of \( U^o_{i,n} \) is difficult due the setup phase. Note that the utilization of a server denotes the fraction of time at least one job is present in the server (including queue). Since a job may not be present in the two-phase server when it executes the setup phase, \( U^o_{i,n} \) is not equal to \( p_{k,n} + q_{k,n} \), which is the sum of the setup phase utilization and the processing phase utilization. We approximate \( U^o_{i,n} \) as \( p_{k,n} + q_{k,n} - (1 - U^r_{k,n})p_{k,n} \), where \( (1 - U^r_{k,n})p_{k,n} \) denotes the fraction of time the server executes the setup phase with no jobs in the queue. Thus, we get \( U^o_{i,n} = q_{k,n}/(1 - p_{k,n}) = X_nv_k^{-1}e_k^0/(1 - X_n\mu_k^{-1}e_k^0) \). The complete AMVA algorithm is given in Algorithm 1.

(b) General service times

We now consider the case when the setup and processing times at two-phase servers and the service time at one-phase servers follow general distributions with known mean and squared coefficient of variation (SCV). We modify the mean sojourn time expressions given in Eqs. (1) and (4) for general service times. The expression of \( S^0_{r,i,n} \) is given as follows

\[ S^0_{r,i,n} = \left( Q^0_{i,n-1} - U^0_{i,n-1} \right) w_i^{-1} + U^0_{i,n-1} + \frac{c^2}{2} w_i^{-1} + w_i^{-1}. \]

(12)

The expression of \( S^r_{k,n} \) is given as follows

\[ S^r_{k,n} = \left( Q^r_{k,n-1} - q_{k,n-1} \right) (\mu_k^{-1} + v_k^{-1}) + q_{k,n-1} \left( \frac{1 + c^2}{2} v_k^{-1} + \mu_k^{-1} \right) + p_{k,n-1} \mu_k^{-1} + v_k^{-1}. \]

(13)
Algorithm 1 AMVA algorithm.

1: Initialize \(Q_{0,n}^0 = 0\) \(i = 1, \ldots, m_n\).
2: Initialize \(Q_{1,n}^0 = 0\) and \(\delta_j(0|n) = 1\) \(j = 1, \ldots, m_1\).
3: Initialize \(Q_{k,n}^0 = 0\) \(k = 1, \ldots, m_k\).
4: for \(n = 1\) to \(N\) do
5: \(\) Compute \(S_{k,n}^n\) using Eq. (1) for \(i = 1, \ldots, m_n\).
6: \(\) Compute \(S_{1,n}^n\) using Eq. (2) for \(j = 1, \ldots, m_j\).
7: \(\) Compute \(X_n\) using Eq. (10).
8: \(\) for \(i = 1\) to \(m_0\) do
9: \(\) \(\) \(Q_{0,i}^n = X_nS_{i,n}^n e_i^T\).
10: \(\) \(\) \(U_{0,i}^n = X_nw_i^{-1}e_i^T\).
11: \(\) \(\) for \(j = 1\) to \(m_j\) do
12: \(\) \(\) \(Q_{j,1}^n = X_nS_{j,n}^n e_j^T\).
13: \(\) \(\) \(U_{j,1}^n = 1 - \delta_j(0|n)\).
14: \(\) \(\) for \(k = 1\) to \(m_k\) do
15: \(\) \(\) Compute \(S_{k,n}^n\) using Eq. (7).
16: \(\) \(\) \(Q_{k,1}^n = X_nS_{k,n}^n e_k^T\).
17: \(\) \(\) \(U_{k,1}^n = X_nv_i^{-1}e_i^T/(1 - X_n\mu_k^{-1}e_i^T)\).
18: \(\) \(\) \(U_{k,1}^n = \sum_{l=1}^{m_k} U_{k,l}^n\).
19: \(\) \(\) Outputs: \(X_n, Q_{0,n}^n, Q_{1,n}^n, Q_{k,n}^n, U_{0,i}^n, U_{1,j}^n, U_{k,l}^n\).

In Eq. (13), the first term, \((Q_{k,n-1}^0 - q_{k,n-1})(\mu_k^{-1} + v_k^{-1})\) denotes the mean time to serve all jobs in the queue (excluding the job in the processing phase), the second term, \(q_{k,n-1}(v_k^{-1} + \mu_k^{-1})\) denotes the mean residual time before a new job enters in the processing phase when the arriving job sees the server in the processing phase, the third term, \(p_{k,n-1}^{-1} = \mu_k^{-1}\) denotes the mean residual time before a new job enters in the processing phase when the arriving job sees the server in the setup phase, and the last term, \(v_k^{-1}\) is the mean processing time. After substituting \(p_{k,n-1}\) by the utilization of the setup phase \((X_{k-1}\mu_k^{-1}e_k^T)\) and \(q_{k,n-1}\) by the utilization of the processing phase \((X_{k-1}v_k^{-1}e_k^T)\) in Eq. (13), and on simplification, we get

\[
S_{k,n}^n = Q_{k,n-1}^0(\mu_k^{-1} + v_k^{-1}) + X_{n-1} \left( \frac{c_{p,k} - 1}{2} \right) v_k^{-1} - 1 + \frac{c_{e,k}}{2} \left( \mu_k^{-1} \right)^2 e_k^T + v_k^{-1}
\]

By following the similar approach as discussed for the exponential service time, we get the expression of \(X_n\) as follows

\[
X_n = \frac{1}{2A} \left( \frac{1}{\alpha_n^2} + 4nA - \alpha_n \right)
\]

where \(A = \sum_{n=1}^{m_n} \sum_{k=1}^{m_k} \frac{c_{e,i}^T}{2} (v_i^{-1}e_i^T)^2 + \frac{c_{e,i}^T}{2} (\mu_i^{-1}e_i^T)^2\). Note that if we set \(c_{e,i}^T = c_{e,k}^T = c_{e,i} = 1\) for \(i = 1, \ldots, m_i\) and \(k = 1, \ldots, m_k\) in Eq. (15), the expression of \(X_n\) reduces to that of given in Eq. (10). The AMVA algorithm is similar to the one with exponential service time except we obtain \(S_{n,n}^n, S_{1,n}^n, X_n\) using Eqs. (12), (14), and (15), respectively.

3.2.2. Network aggregation dis-aggregation approach (NADA)

This section discusses a network aggregation dis-aggregation approach for solving a closed queuing network that consists of two-phase servers, one-phase servers, and load-dependent servers. First, we propose a general approach for an arbitrary network, and later illustrate the approach using a simple example. The approach consists of two steps: network aggregation and network dis-aggregation. In the network aggregation, the large network is iteratively reduced to a smaller size network. In each iteration, a subnetwork is exactly solved, the conditional performance measures are obtained, and the subnetwork is replaced by an equivalent load-dependent server in the original network. In the network dis-aggregation, the aggregated network is iteratively dis-aggregated following the reverse process of the aggregation steps and unconditional performance measures are obtained.

Network aggregation: Consider a closed queuing network consisting of \(M\) servers, where \(M = m_0 + m_1 + m_2\). We first explain the network aggregation procedure for \(M \geq 4\). There are three key steps: initialization, iteration, and termination.

1. Initialization: Let \(c\) denote the iteration number of the NADA method. Set \(c = 1\). We choose any two adjacent servers in the network and denote them by node \(c\) and node \(c + 1\). These servers can be of similar or different types. Next, we aggregate the chosen servers and replace them by an equivalent load-dependent server, \(AGG_c\), in the original network. For aggregation, we follow the flow equivalence principle that is analogous to Norton’s theorem for electric circuits. We short circuit (or remove) all nodes except the chosen servers. This results in a closed queuing subnetwork of node \(c\) and node \(c + 1\). Let \(\lambda_c(n)\) be the throughput of the subnetwork with \(n\) jobs in \(c\)th iteration, where \(n = 1, \ldots, N\) and \(c = 1, \ldots, M - 1\). Then, the service rate at \(AGG_c\) is given as \(\lambda_c(n) = \lambda_n(n)\). We denote the queue length distribution at node \(c\) by a vector \(\Phi_c(n) = (\Phi_{c,0}(n), \Phi_{c,1}(n), \ldots, \Phi_{c,n}(n))\), where \(\Phi_{c,n}(r)\) is the probability that \(r\) jobs are present at node \(c\) in the subnetwork with \(n\) jobs. Note that \(\lambda_c(n), \Phi_{c,0}(n),\) and \(\Phi_{c,1}(n)\) for each \(c\) are exactly obtained by solving the closed queuing subnetwork using Markov chain analysis. For general service times, we fit simple phase-type distributions to the first two moments of the service times (setup and processing times for two-phase servers) and perform an exact Markov Chain analysis (for details, see the online supplement).

2. Iteration: Now the resulting closed queuing network has \(M - 1\) servers. We have completed the first iteration in the initialization step. Therefore, we have \(c = 2\) for the current aggregation. We choose an adjacent server to node \(AGG_{c-1}\), and denote it by node \(c + 1\). Next, we aggregate node \(AGG_{c-1}\) and node \(c + 1\), and replace them by an equivalent load-dependent server, \(AGG_c\). Let \(\lambda_{c+1}(n)\) be the throughput of the subnetwork of \(AGG_{c-1}\) and node \(c + 1\) with \(n\) jobs, where \(n = 1, \ldots, N\). Then, the service rate at \(AGG_c\) is given as \(\lambda_{c+1}(n) = \lambda_n(n)\). Now we set \(c = c + 1\). If \(M - c > 1\), perform next iteration of this procedure, else go to the termination step.

3. Termination: We now perform the last iteration of the aggregation procedure. We have \(c = M - 1\). The original network has reduced to a smaller closed queuing network that consists of two nodes: \(AGG_{c-1}\) and node \(c + 1\) (which is not part of the aggregation). We solve the reduced network with \(N\) jobs and obtain the performance measures. Let \(\lambda_c(N)\) be the throughput of the reduced queuing network. Then, we have the throughput of the original network, \(\lambda_N = \lambda_c(N)\).

If \(M = 3\), we follow the initialization and termination steps. For \(M = 2\), we directly go to the termination step. Based on Chandy, Herzog, and Woo’s (1975) work, we have the following proposition.

Proposition 2. For a closed queuing network consisting of one-phase single-server stations and load-dependent single-server stations with exponential service times, the throughput of the aggregated queuing network is equal to the throughput of the original (non-aggregated) queuing network.
Proposition 2 guarantees that the NADA method is exact for product form queuing networks. When a closed queuing network includes two-phase servers, it does not satisfy the product form properties (Baskett, Chandy, Muntz, & Palacios, 1975). Therefore, the aggregated queuing network is not equivalent to the original network. However, this approach can be used to obtain approximate solutions. From numerical experiments (see in Section 3), it is evident that the approach is fairly accurate in solving a closed queuing network that includes two-phase servers.

Network dis-aggregation: To obtain the performance measures, such as mean queue length and server utilization, we dis-aggregate the aggregated queuing network. For all $c = 3, \ldots, M - 1$, node $AGG_{c-1}$ is an aggregated server of node $AGG_{c-2}$ and node $c$. Therefore, the queue length distributions at node $AGG_{c-2}$ and node $c$ are conditional on the queue length distribution at node $AGG_{c-1}$. Similarly, the queue length distributions at node 1 and node 2 are conditional on the queue length distribution at node $AGG_1$.

Let $\pi_i(r)$ be the probability that $r$ jobs are present at node $i$ in the original (non-aggregated) queuing network, where $i = 1, \ldots, M$ and $r = 0, \ldots, N$. Let $\theta_i(r)$ be the probability that $r$ jobs are present at $AGG_i$ where $j = 1, \ldots, M - 2$ and $r = 0, \ldots, N$. From the termination step ($c = M - 1$), we have $\pi_M(r) = \phi_{M, N}(r)$ and $\theta_{M-2}(r) = \phi_{M,N}(N - r)$ for $r = 0, \ldots, N$.

For $c = M - 2, \ldots, 2$, the queue length distribution at node $c + 1$ and $AGG_c$ are obtained using the principle of conditional probability as follows, for $r = 0, \ldots, N$.

$$\pi_{c+1}(r) = \sum_{n=r}^{N} \phi_{c+1,n}(r)\theta_i(n)$$  \hspace{1cm} (16)

$$\theta_{c-1}(r) = \sum_{n=r}^{N} \phi_{c+1,n}(n - r)\theta_i(n).$$  \hspace{1cm} (17)

For nodes 1 and 2, the queue length distributions can be obtained as follows, for $r = 0, \ldots, N$.

$$\pi_2(r) = \sum_{n=r}^{N} \phi_{2,n}(r)\theta_1(n)$$  \hspace{1cm} (18)

$$\pi_1(r) = \sum_{n=r}^{N} \phi_{1,n}(r)\theta_1(n).$$  \hspace{1cm} (19)

It should be noted that $\phi_{i,0}(n) = 1$ for $i = 1, \ldots, M$, which means if the subnetwork has zero jobs then the probability of having zero jobs at node $i$ is one. From the queue length distributions, we can obtain the mean queue length and the server utilization of node $i$ as $\sum_{r=0}^{\infty} r\pi_i(r)$ and $\sum_{r=0}^{\infty} \pi_i(r)$, respectively.

**Example:** To illustrate the NADA method, we present a simple closed queuing network with $m_1 = 1$, $m_2 = 1$, and $m_3 = 2$ as shown in Fig. 3(a). We consider exponential service times at all servers (including the setup and the processing times at two-phase servers) in the closed queuing network. We set $N = 3$. The service rate at node 1 is given by $r/10$, where $r$ denotes the number of jobs present at node 1. The service rate at node 2 is set to 0.20. The setup and processing rates at node 3 and node 4 are set to 0.20. The visit ratios to node 3 and node 4 are set to 0.50. Fig. 3 illustrates the steps of the NADA method.

**Network aggregation**

1. **Initialization:** In the first iteration, we aggregate node 1 and node 2, and replace them by a load-dependent server, $AGG_1$. We remove all nodes except node 1 and node 2 in the original network and obtain a closed queuing subnetwork as shown in Fig. 4. Let $n$ be the number of jobs in the subnetwork. For each $n = 1, \ldots, 3$, we solve the subnetwork using Markov chain analysis and obtain $\lambda_1(n)$, $\Phi_1(n)$, and $\Phi_2(n)$. Note that we have $\lambda_1(1) = \lambda_1(r)$. We get $\lambda_1(1) = 0.0667$, $\lambda_1(2) = 0.12$, $\lambda_1(3) = 0.1579$, $\Phi_1(1) = [0.3333, 0.6667]$, $\Phi_2(1) = [0.6667, 0.3333]$, $\Phi_1(2) = [0.2, 0.4, 0.4]$, $\Phi_2(2) = [0.4, 0.4, 0.2]$, $\Phi_1(3) = [0.1579, 0.3158, 0.3158, 0.2105]$, and $\Phi_2(3) = [0.2105, 0.3158, 0.3158, 0.1579]$.

2. **Iteration:** In iteration 2, we have a closed queuing network with 3 servers (see Fig. 3b). Now we aggregate node 3 and $AGG_1$, and replace them by $AGG_2$. To obtain the service rate at $AGG_2$, we solve a subnetwork as shown in Fig. 5. Note that the service rate at $AGG_2$ is equal to the throughput at point A in the subnetwork. We denote the service rate at $AGG_2$ by $\lambda_2(r)$, $r = 1, \ldots, 3$. Due to the different visit ratios, the throughput at $AGG_1$ is not equal.
Table 2
Experiment design for exponential service times (time in seconds).

<table>
<thead>
<tr>
<th>$(μ₁^{−1}, ν_1^{−1})$</th>
<th>$(μ₂^{−1}, ν₂^{−1})$</th>
<th>$w₁^{−1}(r)$</th>
<th>$w₂^{−1}$</th>
<th>N</th>
<th>Number of scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3, 3), (3, 5), (5, 3)</td>
<td>(3, 3), (3, 5), (5, 3)</td>
<td>3</td>
<td>r</td>
<td>4</td>
<td>1,3,5,7,9,11,13,15</td>
</tr>
</tbody>
</table>

Table 3
Experiment design for general service times (time in seconds).

<table>
<thead>
<tr>
<th>$(μ₁^{−1}, ν_1^{−1}, ν₂^{−1}, ν₃^{−1})$</th>
<th>$(μ₂^{−1}, ν₂^{−1}, ν₃^{−1})$</th>
<th>$w₁^{−1}(r)$</th>
<th>$w₂^{−1}$</th>
<th>N</th>
<th>Number of scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3,0,4,3,0.4)</td>
<td>(3,0,4,3,0.4)</td>
<td>3</td>
<td>r</td>
<td>4</td>
<td>0.33</td>
</tr>
<tr>
<td>(3,0,8,5,0,8)</td>
<td>(3,0,8,5,0,8)</td>
<td>3</td>
<td>r</td>
<td>4</td>
<td>0.33</td>
</tr>
<tr>
<td>(5,1,2,3,1,2)</td>
<td>(5,1,2,3,1,2)</td>
<td>3</td>
<td>r</td>
<td>4</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Table 4
Summary of absolute errors in the performance measures ($X_0$ in jobs/minute).

<table>
<thead>
<tr>
<th>Method</th>
<th>Statistics</th>
<th>$X_0$</th>
<th>$Q_1^{N−1}$</th>
<th>$Q_2^{N−1}$</th>
<th>$Q_3^{N−1}$</th>
<th>$Q_4^{N−1}$</th>
<th>$U_1^{N−1}$</th>
<th>$U_2^{N−1}$</th>
<th>$U_3^{N−1}$</th>
<th>$U_4^{N−1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMVA</td>
<td>Mean</td>
<td>0.03</td>
<td>0.00</td>
<td>0.31</td>
<td>0.16</td>
<td>0.15</td>
<td>0.02</td>
<td>0.00</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>Minimum</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>0.10</td>
<td>0.00</td>
<td>0.84</td>
<td>0.65</td>
<td>0.70</td>
<td>0.03</td>
<td>0.01</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>NADA</td>
<td>Mean</td>
<td>0.02</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Minimum</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>0.10</td>
<td>0.01</td>
<td>0.09</td>
<td>0.11</td>
<td>0.07</td>
<td>0.03</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>General service times</td>
<td>Mean</td>
<td>0.08</td>
<td>0.00</td>
<td>0.18</td>
<td>0.12</td>
<td>0.12</td>
<td>0.03</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>Minimum</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>0.45</td>
<td>0.02</td>
<td>1.25</td>
<td>0.62</td>
<td>0.61</td>
<td>0.04</td>
<td>0.03</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>NADA</td>
<td>Mean</td>
<td>0.34</td>
<td>0.02</td>
<td>0.16</td>
<td>0.13</td>
<td>0.11</td>
<td>0.01</td>
<td>0.02</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>Minimum</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>0.69</td>
<td>0.03</td>
<td>0.87</td>
<td>0.44</td>
<td>0.43</td>
<td>0.02</td>
<td>0.05</td>
<td>0.04</td>
<td>0.04</td>
</tr>
</tbody>
</table>

3.3. Numerical validation of the solution methods

In this section, we validate the solution methods developed in the previous section. We consider the queueing network shown in Fig. 3 for the validation purpose. The queueing network comprises a load-dependent server, one-phase server, and two-phase server with both parallel and series configurations of nodes. We generate several scenarios by varying the input parameters. Tables 2 and 3 show experiment designs for exponential and general service times, respectively. For each scenario, we obtain estimates of the performance measures (throughput, mean queue length, and utilization) using AMVA, NADA, and simulation model. As shown in Fig. 3, in the NADA method, node 1 and node 2 are aggregated first, followed by node 3 and node 4. We implement the analytical methods (AMVA and NADA) in Matlab software. The simulation model is built in Arena software (www.arenasimulation.com). The flow chart of the simulation model is given in the online supplement. We measure the accuracy of the solution methods by calculating absolute errors in the performance measures, $(|Y_0 - Y_1|)$. where $Y_0$ and $Y_1$ are the analytical and simulation results, respectively. Table 4 summarizes the absolute errors in the performance measures for exponential and general service times. Results indicate that the solution methods are quite accurate in estimating the performance measures. However, concerning maximum errors in the estimates, the NADA method is comparatively more accurate than the AMVA method. In the AMVA method, the maximum absolute error (across all measures) is 0.84 with exponential service time and 1.25 with general service time. In the NADA method, the maximum absolute error is 0.21 with exponential service time and 0.87 with general service time.

3.3.1 Accuracy of the solution methods with varying $N$

We compare the accuracy of the solution methods by varying the number of jobs in the network. The input parameters and the experiment design are taken from Tables 2 and 3. In Fig. 6, we report the absolute errors in the inter-departure time for both expo-
3.3.2. Order of aggregation in the NADA method

The order of aggregation in the NADA method may impact its accuracy. To understand the impact of the aggregation order, we obtain the estimates of the performance measures with different aggregation orders and compare them with results obtained using the simulation model. Table 5 shows the estimates of the performance measures with different input parameters and aggregation orders.

An aggregation order ‘1-2-3-4’ indicates that nodes 1 and 2 are aggregated first, followed by node 3 and node 4. Our analysis shows that first aggregating non two-phase servers, that is, load-dependent servers or one-phase servers, results in more accurate estimates of the performance measures. Aggregation orders ‘1-2-3-4’ and ‘1-2-4-3’ give estimates close to those obtained with the simulation model. We find that for a large N, the estimates are more close to those obtained with the simulation model. We also analyze the impact of the aggregation order for product form queuing networks. We consider exponential service times and set the mean setup times (μ₁ and μ₂) to zero for the two-phase servers to convert the network shown in Fig. 3 to a product form queuing network. We report the results of our analysis in the online supplement. We find that the NADA method is exact for the product form networks and the aggregation order does not impact the accuracy of the solution methods. These results reaffirm Proposition 2.

3.3.3. Computational efficiency

We now compare the computational efficiency of the proposed AMVA method with Van Doremalen’s (1986) method. From Algorithm 1, we see that our method has a time complexity of O(MN²). We show Van Doremalen’s (1986) method in the online supplement. This method, in addition, follows an iterative approximation for each value of n to compute Xₙ. Therefore, the number of iterations required in Van Doremalen’s (1986) method is always more or equal to one required in our approach. We also conduct numerical experiments to confirm these claims. Table 6 shows the number of iterations required to get the performance measures for different values of the input parameters. Results indicate that our method always requires a smaller or equal number of iterations to estimate the performance measures.

In the online supplement, we discuss the run time complexities of the proposed AMVA and Van Doremalen’s (1986) method and show their computational times for large problem instances.

4. Modeling of parallel process flows in an automated container terminal

In this section, we present an application of the proposed modeling approach in automated container terminals and show its benefits in comparison to the existing sequential modeling approach.

4.1. System description and parallel process flows

The seaside operations at an automated container terminal are carried out by three key resources: quay cranes (QCs), AGVs, and automated stack cranes (ASCs). We describe the unloading or import operation of containers from a berthed vessel. The flow of processes is shown in Fig. 7. The container loading or export operation is similar to the import operation, but the flows are in re-
verse direction of the unloading operation. In the container unloading operation, a QC work cycle includes container pick-up from a berthed vessel, movement of the QC's trolley from the pick-up location in the vessel to the drop-off location, positioning (drop-off) of the container on an AGV, and movement of the trolley back to the container pick-up location. The loaded AGV travels along a common guide path to a preassigned stack block. At the stack block, an ASC picks-up the container from the AGV, carries it to the storage location, positions over there, and the empty trolley moves back to the pick-up location. As soon as the ASC picks-up

the container, the AGV leaves the stack block and travels back to the quayside for loading the next container.

In the quayside operation, while the AGV is traveling from the stackside to the quayside, the QC can simultaneously fetch a container from the vessel. The movement of the AGV from the stackside to the quayside and the container fetching from the vessel are two independent processes which are performed simultaneously (in parallel). Once the AGV arrives at the quayside, the QC loads the container on the AGV. This process requires synchronization of the AGV and the QC. After the QC loads the container on the
AGV, the QC retrieves the next container, and the AGV travels to the stackside. In the stackside operation, an ASC's work cycle involves the following steps: (i) the ASC's trolley picks-up container from an AGV, (ii) travels to the storage location in the stack block and stores the container, and (iii) travels back to the pick-up location. While an AGV is traveling from the quayside to the stackside, the ASC can simultaneously perform steps (ii) and (iii). To perform step (i), the ASC must be synchronized with the AGV. Once the ASC picks-up the container, the AGV travels to the quayside and brings the next container, and the ASC performs the storage operations.

We consider a container terminal layout that consists of $N_q$ single-trolley QCs, $N_s$ ASCs, and $N$ AGVs. Fig. 8 shows a container terminal with 2 QCs and 8 ASCs. We denote the $i$th QC and the $j$th ASC by $Q^i_c$ and $A^{j}_c$, respectively, where $i = 1, \ldots, N_q$ and $j = 1, \ldots, N_s$. An arriving AGV waits at the quayside buffer area when the QC is busy in the container fetching operations or serving the AGVs in front of it. We assume that an AGV can carry only one standard size container (1 TEU) at a time. The destination of a loaded AGV is uniformly distributed across all stack blocks. Similarly, an empty AGV has an equal probability of being assigned to any of the QCs. Each stack block has a single-trolley ASC and serves the AGVs in first come first basis. The QCs and ASCs dwell after finishing their setup tasks, i.e., container fetching and storage operations of the container, respectively. The dwell point of a QC is the container drop-off location at the quayside. An ASC dwells at the container pick-up point at the stackside. The throughput capacity of a container terminal is a critical decision parameter at the design phase of the container terminal. Thus, we establish the maximum throughput capacity of the container terminal; therefore, assume that AGVs do not dwell and continuously transport containers from the quayside to the stackside.

4.2. Two-phase server based closed queuing network

In this section, we develop a queuing model for the container unloading operations at an automated container terminal with AGVs. The terminal layout is shown in Fig. 8. The yard area is connected to the quay area by two parallel paths$^1$. In the unloading process, an AGV receives service (loads a container) at a QC, travels to an ASC, receives service (hands over the container) at an ASC, travels back to a QC, and repeats the cycle. We model the cyclic process of AGVs by a closed queuing network, where the AGVs represent jobs and QCs, ASCs, and travel paths represent service stations. As explained earlier, a part of the service operations at the QCs and the ASCs can be executed in the absence of the AGVs. Thus, we model the QCs and the ASCs as two-phase servers.

The setup phase of the two-phase server corresponds to a QC denotes the container fetching process from the vessel and the processing phase denotes the container loading on the AGV. Similarly, the setup phase of the two-phase server corresponds to an ASC denotes the storage process of a container in the stack block by the ASC and the processing phase denotes the container unloading process from the AGV. Fig. 9 shows the two-phase server based closed queuing network corresponding to the layout in Fig. 8. We denote the mean processing and setup times of the $i$th two-phase server corresponds to QC by $v_{qi}^{-1}$ and $\mu_{qi}^{-1}$, respectively. The SCV of the setup and processing times of QC is denoted by $c_{qi}^{2}$, respectively. Node 4 to 11 are the two-phase servers representing ASC to ASC. We denote the mean processing and setup times of $j$th two-phase server corresponds to ASC by $v_{sj}^{-1}$ and $\mu_{sj}^{-1}$, respectively. The SCV of the setup and processing times of ASC are

---

$^1$ One example of such a terminal layout is the APM terminal at the JNPT port, Mumbai, India (www.apmtmumbai.com).
denoted by $c^i_j$ and $c^j_i$, respectively. The AGVs transport containers from the quayside to the stackside and travel along a common guide path. We assume that AGVs travel at a uniform speed $v_h$. We model the common paths as delay servers. Node 3 and Node 12 are two delay servers representing the travel times from the quayside to the stackside and from the stackside to the quayside, respectively. The mean service times of Node 3 and Node 12 are denoted by $u_{q3}^{-1}$ and $u_{s3}^{-1}$, respectively. Let $L_{q3}(L_{s3})$ be the mean length of the travel path from the quayside (stackside) and the stackside (quayside). Then, we get $u_{q3}^{-1} = L_{q3}/v_h$ and $u_{s3}^{-1} = L_{s3}/v_h$.

### 4.3. Validation of the analytical model

We use the NADA method, developed in the previous section, for solving the closed queuing network shown in Fig. 9. The choice of the solution method is solely determined by its high accuracy (in terms of maximum error) compared to the AMVA method (see Section 3.3). The design of numerical experiments is shown in Table 8. We vary the number of QCs ($N_q$), the number of ASCs ($N_s$), and the number of AGVs ($N$) to generate several scenarios. The NADA method is implemented in Matlab software, and the performance measures (throughput, mean queue length, and

![Fig. 9. Closed queuing network for the seaside operations at an automated container terminal with 2 QCs and 8 ASCs.](image-url)
utilization) are obtained for each scenario. Based on the insights obtained in Section 3.3.2, we aggregate the queuing network as follows: node 1 and node 2 are aggregated first, followed by node 12, node 4 to node 15, and at last node 3. Table 7 shows the input parameters\(^2\). The procedure to obtain \(L_{q1}\) and \(L_{sq}\) is given in the online supplement.

The results obtained using the analytical model are validated against simulation results. We build a simulation model in Arena software. The flow chart of the simulation model is shown in the online supplement. For each scenario, we set 1 day as the warm-up period and then run 10 replications of 10 days each in the simulation model. We measure the accuracy of our analytical model using absolute errors. A summary of absolute errors in throughput, mean queue lengths, and utilization is shown in Table 9. The numerical experiment results show that the analytical model gives quite accurate estimates of the performance measures in comparison to the simulation model.

4.4. Insights

4.4.1. Modeling approach and estimates of the terminal performance measures
In this section, we understand the impact of modeling approach on the estimates of the terminal performance measures. Existing studies on container terminals adopt the sequential modeling approach. We compare the throughput of a container terminal obtained using the proposed (parallel) modeling approach with one obtained using the sequential modeling approach. We consider two terminal configurations: a small container terminal (2 QCs, 8 ASCs) and a large container terminal (4 QCs, 16 ASCs). Each terminal is modeled using the parallel and sequential modeling approaches, and the throughput is obtained. For the parallel modeling approach, we obtain the terminal throughput using the proposed analytical model (denoted by \(X_{par,ana}\)) and using the simulation model (denoted by \(X_{par,sim}\)). For the sequential modeling approach, we obtain the throughput using a simulation model and denote it by \(X_{seq,sim}\). The input parameters are taken from Table 7. In the sequential modeling approach, the container fetching and loading operations are carried out in a sequence. Therefore, the service time of a QC (similarly for an ASC) is equal to the sum of its setup and processing times. Fig. 10 shows the throughput (containers/hr) obtained using the parallel and sequential modeling approaches. We observe that approximating the parallel process flows in the sequential modeling approach underestimates the container terminal throughput on average by 28% and at maximum by 47%. However, our analytical solution approach, when applied for the container terminal operations, estimates the throughput quite accurately with maximum 5% gap (\(|X_{par,sim} - X_{par,ana}| \times 100/X_{par,sim}\)). These error values may depend upon the input parameters, such as layout, but nevertheless, the parallel modeling approach always outperforms the sequential modeling approach.

We find that the benefit of parallel operations is higher for the large system. For instance, we get 47% (maximum) improvement in throughput of a container terminal with 4 QCs and 16 ASCs compared to 36% (maximum) improvement in the throughput of a container terminal with 2 QCs and 8 ASCs. Also, the benefit of parallel operations is impacted by the utilization of QCs and ASCs. Thus, we understand the impact of QC and ASC utilization on the

---

\(^2\) The parameters are obtained through discussions with a terminal service provider. These parameters may vary based on the terminal configuration. However, our approach remains valid.

---

**Table 9**

Summary of absolute errors in performance measures for the container terminal model.

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Throughput (containers/hour)</th>
<th>Mean queue length at</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>QC</td>
<td>ASC</td>
</tr>
<tr>
<td>Mean</td>
<td>1.39</td>
<td>0.06</td>
<td>0.01</td>
</tr>
<tr>
<td>Minimum</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Maximum</td>
<td>3.58</td>
<td>0.21</td>
<td>0.05</td>
</tr>
</tbody>
</table>

---

**Fig. 10.** Throughput (container/hour) with the parallel and sequential modeling approaches.
throughput gap \(X_{\text{par,sim}} - X_{\text{seq,sim}}\). As expected, the benefit of the parallel modeling approach decreases as the utilization increases (see Table 10). With increasing utilization of the two-phase servers (or QCs and ASCs), the probability that the setup task has been completed prior to a job (AGV) arrival decreases. At a highly utilized resource, the total service requirement of an arriving job is nearly equal to the sum of the setup and processing times, which is equivalent to the service requirement in the sequential modeling approach. Keeping other parameters constant, for a large system (for example, 4 QCs, 16 ASC), the utilization of QCs/ASCs is lower compared to a small system (2 QCs, 8 ASCs). Therefore, the benefit of parallel modeling is higher in large systems.

### 4.4.2. Impact of quasiady container fetching and loading times on the terminal throughput

At the quayside, a QC performs two operations: container fetching and loading of the container on an AGV. The time required to perform these operations may impact the throughput of the terminal. Note that the container fetching time of a QC \(\mu_{q}^{-1}\) depends on several parameters, such as outreach, backreach, lift height, hoisting speed, and trolley speed of the QC. For example, a QC with low hoisting and trolley speeds will require more time in fetching a container. Similarly, the loading time of a container on an AGV \(v_{q}^{-1}\) depends on the precision level of the QC and the synchronization between the QC and the AGV. Thus, it is important to understand the impact of \(\mu_{q}^{-1}\) and \(v_{q}^{-1}\) on the terminal throughput. We consider two container terminal configurations and obtain the throughput with different values of \(\mu_{q}^{-1}\) and \(v_{q}^{-1}\) (see Fig. 11). We vary \(\mu_{q}^{-1}\) from 10 to 190 seconds and \(v_{q}^{-1}\) from 5 to 15 seconds for both terminal configurations. Results indicate that as \(\mu_{q}^{-1}\) increases, the throughput decreases in a non-linear fashion. An increment from the lower value of \(\mu_{q}^{-1}\) does not reduce the throughput as much as it reduces when the same increment happens at the higher values of \(\mu_{q}^{-1}\). For instance, at \(v_{q}^{-1} = 5\) seconds, it results in 2.55% reduction in the throughput. Whereas, if \(\mu_{q}^{-1}\) increases from 70 to 90 seconds, it results in 15.67% reduction in the throughput. However, if \(\mu_{q}^{-1}\) increases from 170 to 190 seconds, it results in 10.50% reduction in the throughput. When the setup time \(\mu_{q}^{-1}\) is less, the probability that the setup is executed before the arrival of an AGV at the quay crane is higher. Thus, a slight increment in \(\mu_{q}^{-1}\) from the low value does not have much impact on the throughput. In Fig. 12, we show % gap in throughput obtained using parallel and sequential modeling approaches with varying \(\mu_{q}^{-1}\) and \(v_{q}^{-1}\). We find that the gap diminishes with increasing \(\mu_{q}^{-1}\). As shown in Table 10, the utilization of ASCs is quite low. Therefore, the increase in \(\mu_{s}^{-1}\) will not impact the throughput obtained using the parallel modeling approach. However, in the sequential modeling approach, the throughput decreases when \(\mu_{s}^{-1}\) increases. Consequently, the gap diminishes with \(\mu_{s}^{-1}\). Note that we get a different pattern in % gap (see Fig. 12) when we vary \(\mu_{q}^{-1}\) and \(v_{q}^{-1}\) due to the higher utilization of QCs.

### 5. Modeling of parallel process flows in a compact storage system

In this section, we present another application of our modeling approach in a shuttle-based compact storage system. We show the performance gap between our approach and the existing approach when both approaches are applied to the same system.

#### 5.1. System description and parallel process flows

Shuttle-based compact storage systems are automated warehousing systems, known for their flexibility and high space-usage efficiency (Tappia et al., 2017). Such systems are popularly used for storage and handling of unit-loads in e-commerce, textile, and

| Number of 
| Number of 
| 4 QCs, 8 ASCs | 4 QCs, 16 ASCs |
| --- | --- | --- | --- |
| % gap in throughput | % utilization of | % gap in throughput | % utilization of |
| \(\frac{(X_{\text{par,sim}} - X_{\text{seq,sim}})^{100}}{X_{\text{seq,sim}}^{100}}\) | QCs | ASCs | QCs | ASCs |
| 4 | 36.12 | 57.10 | 7.80 | 47.20 | 15.90 | 3.00 |
| 6 | 20.78 | 74.80 | 9.60 | 41.76 | 29.00 | 5.00 |
| 8 | 11.04 | 82.70 | 10.40 | 36.14 | 41.40 | 6.60 |
| 10 | 5.47 | 85.60 | 10.50 | 30.28 | 52.20 | 7.70 |
| 12 | 3.26 | 88.30 | 10.60 | 24.58 | 59.90 | 8.70 |
| 14 | 2.12 | 89.50 | 10.50 | 19.48 | 66.70 | 9.50 |
| 16 | 1.84 | 91.90 | 11.20 | 15.42 | 72.30 | 10.20 |
| 18 | 1.25 | 92.70 | 11.60 | 12.09 | 75.10 | 10.30 |

### 4.4.3. Impact of stackside container storage and pick up times on the terminal throughput

We now obtain the terminal throughput with different values of the container storage time in the stack block \(\mu_{s}^{-1}\) and the container pick-up time from an AGV by the ASC \(v_{s}^{-1}\). Note that \(\mu_{s}^{-1}\) depends upon the stack block configuration (number of tiers, number of bays, and number of rows) and ASC parameters, such as the travel speed of gantry, hoisting speed, and trolley speed. We consider two container terminal configurations: 2 QCs, 8 ASCs, 8 AGVs and 4 QCs, 16 ASCs, 16 AGVs. We obtain the terminal throughput by varying \(\mu_{s}^{-1}\) from 50 to 500 seconds and \(v_{s}^{-1}\) from 5 to 35 seconds (see Fig. 13).

Results indicate that as \(\mu_{s}^{-1}\) increases, the throughput decreases. For low values of \(\mu_{s}^{-1}\), the throughput remains nearly constant. However, for large values (more than 200 seconds), the throughput linearly decreases as \(\mu_{s}^{-1}\) increases. Also, as \(v_{s}^{-1}\) increases, the throughput decreases. One possible explanation for this non-linear pattern in the throughput with varying \(\mu_{s}^{-1}\) is that when the setup time \(\mu_{s}^{-1}\) is small, the probability that the setup is executed before the arrival of an AGV at the stack crane is high. Thus, a slight increment in \(\mu_{s}^{-1}\) from the low value does not have much impact on the throughput. In Fig. 14, we show % gap in throughput obtained using parallel and sequential modeling approaches with varying \(\mu_{s}^{-1}\) and \(v_{s}^{-1}\). We find that the gap diminishes with increasing \(\mu_{s}^{-1}\). As shown in Table 10, the utilization of ASCs is quite low. Therefore, the increase in \(\mu_{s}^{-1}\) will not impact the throughput obtained using the parallel modeling approach. However, in the sequential modeling approach, the throughput decreases when \(\mu_{s}^{-1}\) increases. Consequently, the gap diminishes with \(\mu_{s}^{-1}\). Note that we get a different pattern in % gap (see Fig. 12) when we vary \(\mu_{q}^{-1}\) and \(v_{q}^{-1}\) due to the higher utilization of QCs.
fresh produce warehouses (Azadeh, De Koster, & Roy, 2019). Fig. 15 shows a typical shuttle-based compact storage system. A shuttle-based compact storage system can have a single-tier or multi-tiers. A single-tier shuttle-based compact storage system comprises a storage tier, multiple shuttles, and a transfer car (see Fig. 16(a)). The storage tier consists of a cross-aisle in the middle of the tier and a set of multi-deep lanes on both sides of the cross-aisle. The cross-aisle orthogonally divides the lanes into two parts. At one end of the cross-aisle, loading/unloading (l/u) point is located. Each lane has several storage places (columns) to hold unit loads of a product. A fleet of shuttles and a transfer car provide horizontal movements to unit loads within the tier. The transfer car and shuttles are restricted to travel within the cross-aisle and lanes, respectively. However, to change the shuttle’s lane, the transfer car carries the shuttle through the cross-aisle. Thus, a shuttle can access any lane within the tier. As discussed earlier, a shuttle and the transfer car involve parallel movements in processing a retrieval or storage transaction. While in literature, these movements are assumed to be sequential, we apply the modeling approach discussed in Section 3 and incorporate these parallel movements in our analytical model. We consider a single-tier compact storage system. Our choice of the single-tier system is based on the following. The storage tier area consists of the resources which involve the parallel movements. The queuing model of a single-tier system is the foundation for the multi-tier system’s model. We show the detailed queuing network model of a single-tier compact storage system in the online supplement. Using the queuing model, we obtain the performance measure (throughput) of a single-tier compact storage system and compare it with the throughput obtained using the sequential modeling approach (Tappia et al., 2017).
Fig. 13. Throughput with varying the setup time and the processing time of ASCs.

Fig. 14. Percentage gap in throughput (% underestimation of throughput in sequential modeling) with varying the setup time and the processing time of ASCs.

Fig. 15. (a) An illustration of a shuttle-based compact storage system (source: Total Solution Provider Group), (b) a shuttle, and (c) a transfer car (source: Automha).
5.2. Impact of the modeling approach on the system throughput

We now show the impact of assuming sequential movements of the shuttles and the transfer car on the throughput capacity. We select two designs with depth/width (D/W) ratio equal to 1.0 and 2.0. The number of storage positions are fixed to 10,000. The service time expressions and other input parameters are given in the online supplement. Fig. 17 shows the throughput obtained in the parallel and sequential modeling approaches with varying number of shuttles. We have the following notations: $X_{par,ana}$ denotes the throughput with parallel modeling approach using analytical model, $X_{par,sim}$ denotes the throughput with parallel modeling approach using a simulation model, and $X_{seq,sim}$ denotes the throughput with sequential modeling approach using a simulation model. We observe that the sequential modeling approach underestimates the system throughput. The maximum difference in throughput, $|X_{seq,sim} - X_{par,sim}|$ is 9%. However, with increasing utilization of the transfer car, the throughput difference decreases (see Table 11). When the utilization of the transfer car is high, there will be at least one shuttle waiting for the transfer car before it drops off the on board shuttle. Therefore,

<table>
<thead>
<tr>
<th>No. of shuttles</th>
<th>D/W=1</th>
<th>D/W=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_{seq,sim} - X_{par,sim}$</td>
<td>82.00%</td>
<td>72.70%</td>
</tr>
<tr>
<td>$X_{seq,sim} - X_{par,sim}$</td>
<td>99.20%</td>
<td>96.80%</td>
</tr>
<tr>
<td>$X_{seq,sim} - X_{par,sim}$</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

Table 11: Difference in throughput (transactions/hour) obtained using parallel and sequential modeling.
there is little parallel processing between a shuttle and the transfer car. In fact, the transactions are first processed by a shuttle and then by the transfer car, which is similar to the sequential operations.

6. Conclusions

This paper presents a stochastic modeling approach for parallel process flows, which involves simultaneous operations of multiple resources while processing a job. These processes are commonly found in several intra-logistics systems, such as container terminals, distribution warehouses, and cross-docks. Specifically, we model the simultaneous operations in which hard-coupling of resources is essential to hand over the unfinished jobs. To model such operations, we propose a two-phase server based closed queueing network. A closed queueing network is particularly useful in determining the (maximum) throughput capacity of a system. To solve the resulting network, we develop two solution methods. The first method is an approximate mean value analysis algorithm, and the second method is based on the flow-equivalence. Both methods are validated by numerical experiments. We derive insights on the accuracy of the solution methods.

To demonstrate the parallel modeling approach, we consider two widely studied intra-logistics systems: an automated container terminal with AGVs and a shuttle-based compact storage system. In the case of the container terminal, we develop a closed queueing network with two-phase servers for the seaside container unloading operations. While the existing studies assume sequential operations of the QCS/ASCs and AGVs, our model incorporates the simultaneous operations of the QCS/ASCs and AGVs. We compare the performance measures obtained using the parallel modeling approach with the existing sequential modeling approach. The results show that the sequential modeling approach for an automated container terminal can underestimate the throughput capacity substantially. In our parameter setting, we found the difference in the throughput capacity up to 47%. The proposed parallel modeling approach is quite accurate with 5% maximum gap in the throughput. However, we find that the benefit of the parallel modeling approach decreases as the utilization of QCS/ASCs increases. We show another application of our modeling approach in the shuttle-based compact storage system. We consider a single-tier compact storage system and estimate the throughput capacity using the parallel modeling approach. Results show that using the sequential modeling approach used for the compact storage system results in underestimation of the throughput capacity up to 9%. In the future, the proposed modeling approach can be applied to model other systems where resources work simultaneously, for example, automated warehousing systems and healthcare systems.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2020.08.006.

References