

Recursive decomposition/aggregation algorithms for performance metrics calculation in multi-level assembly/disassembly production systems with exponential reliability machines

Yishu Bai & Liang Zhang

To cite this article: Yishu Bai & Liang Zhang (2023) Recursive decomposition/aggregation algorithms for performance metrics calculation in multi-level assembly/disassembly production systems with exponential reliability machines, International Journal of Production Research, 61:23, 8133-8158, DOI: [10.1080/00207543.2023.2166622](https://doi.org/10.1080/00207543.2023.2166622)

To link to this article: <https://doi.org/10.1080/00207543.2023.2166622>



Published online: 27 Jan 2023.



Submit your article to this journal [↗](#)



Article views: 362



View related articles [↗](#)



View Crossmark data [↗](#)



Citing articles: 3 View citing articles [↗](#)



Recursive decomposition/aggregation algorithms for performance metrics calculation in multi-level assembly/disassembly production systems with exponential reliability machines

Yishu Bai and Liang Zhang

Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, USA

ABSTRACT

Developing accurate and computationally efficient algorithms for system performance metrics calculation is critical to implementing effective control and optimization in manufacturing system operations. In this paper, we propose a recursive decomposition/aggregation-based method for calculating the performance metrics of assembly/disassembly systems with multiple merge/split operations and sub-assemblies. It is assumed that the machines follow the exponential reliability model and the buffers are of finite capacity. To achieve this, we first consider assembly systems with multiple component lines merging at a single assembly operation. By decomposing the system into a set of virtual serial lines, we derive an analytical procedure to approximate the starvation and blockage probabilities of the merge operation, which are used to recursively update the parameters of the virtual serial lines. Then, the performance metrics of the original assembly system are approximated based on the corresponding machines and buffers in these virtual serial lines. Next, we extend the algorithm to assembly/disassembly systems with multiple merge/split operations and sub-assemblies. This is accomplished by identifying the so-called assembly/disassembly units formed based on the virtual serial lines and applying the calculations derived earlier recursively. Simulation experiments are carried out to justify the convergence, computational efficiency, and approximation accuracy of the proposed algorithms. An industrial case study is presented to demonstrate the theoretical methods in practical applications.

ARTICLE HISTORY

Received 25 September 2021
Accepted 25 October 2022

KEYWORDS

Performance evaluation;
assembly systems;
disassembly systems;
exponential machines;
bottleneck analysis

1. Introduction

1.1. Background

Being able to calculate the performance metrics of a production system (e.g. throughput, work-in-process) based on the parameters of the workstations (e.g. mean time between failures and mean time to repair) and buffers involved is critical in manufacturing systems research and practice. It provides the necessary foundation to quantitatively evaluate the effects of potential continuously improvement projects, system modifications, and designs as well as for parameter optimization, without directly experimenting on the physical systems. For production systems, two approaches are commonly used for the purpose of performance metrics calculation: discrete-event simulation and analytical formula-based algorithms. The former is usually conducted via a computer program that simulates the evolution of the system states over time based on the system's mathematical model and statistically calculates the performance

metrics. Nowadays, there are many powerful general-purpose commercial software developed for discrete-event simulation such as Arena, Simul8, FlexSim, and SimEvents, which offer great versatility, connectivity, results visualisation capability, a wide range of supporting functionalities, and have been successfully applied in numerous industrial applications (see, for instance, Negahban and Smith 2014; Neeraj et al. 2018; Omogbai and Salonitis 2016; Kumar, Mahesh, and Kumar 2015; Shukla, Soni, and Kumar 2021). On the other hand, however, since a production system is typically modeled as a stochastic queueing system, the simulation typically has to run for a sufficiently long period of time and/or for a sufficient number of replications to ensure certain statistical precision. This time-consuming issue is commonly recognised as one of the main shortcomings of the simulation-based approach (see in Mourtzis, Doukas, and Bernidaki 2014; Fowler and Rose 2004). This problem is more pronounced in simulation-based optimization tasks since almost all of the CPU time required is

used by simulation experiments to evaluate the quality of candidate solutions, as reported in Gansterer, Almeder, and Hartl (2014), Yegul et al. (2017), Spieckermann et al. (2000), and Tekin and Sabuncuoglu (2004).

To counter the computational efficiency issue of the discrete-event simulation approach and to explore the internal structural properties of production systems, developing accurate and computationally efficient, analytical formula-based algorithms for production system performance metrics calculation becomes an important area in manufacturing systems research (see, for instance, Dallery and Gershwin 1992; Papadopoulos and Heavy 1996; Li and Meerkov 2008). Below, we briefly review the results-to-date on analytical algorithms for performance metrics calculation in production systems.

1.2. Literature review

As the most fundamental and commonly used topology, the serial production line model has been studied extensively (see, for instance, Dallery, David, and Xie 1989; Jacobs and Meerkov 1995; Yao 1994; Li et al. 2009; Zhang et al. 2013; Bai et al. 2020). On the other hand, the studies on assembly systems, which are also commonly seen in manufacturing practice, are relatively limited. This paper is devoted to the study of performance metrics calculation for assembly systems with finite capacity buffers and unreliable machines following the exponential reliability model.

In the current literature, the problem of performance metrics calculation in assembly systems is usually studied based on the results of serial production lines. Typically, researchers use a number of serial production lines to represent/approximate an assembly system and apply certain serial line analysis approaches accordingly. Since the majority of serial line analysis and analytical performance calculation methods in the literature can be grouped into two categories – decomposition and aggregation, we review the results-to-date for assembly systems also based on this classification.

Based on the decomposition technique for performance evaluation in serial production lines developed by Dallery, David, and Xie (1989), representative papers on assembly system analysis can be found in Liu and Buzacott (1990), Gershwin (1991), Mascolo, David, and Dallery (1991), Jeong and Kim (1997), Gershwin and Burman (2000), Xia et al. (2016), Wang et al. (2018), Manitz (2008), and Manitz (2015). Specifically, early studies in this area start from analyzing homogeneous (also called *synchronous*) assembly/disassembly (A/D) systems with geometric reliability machines in Liu and Buzacott (1990) and Gershwin (1991), and exponential machines in Mascolo, David, and Dallery (1991). These

methods can be extended to a non-homogeneous (also called *asynchronous*) system by transforming it into a homogeneous one. Direct analysis of non-homogeneous assembly systems is carried out by Jeong and Kim (1997), Helber (1998), and Gershwin and Burman (2000), where the machines' mean times between failures, mean times to repair, and processing times are assumed to be exponentially distributed. Nonetheless, Jeong and Kim (1997) mentions that the method proposed therein may not have a higher accuracy when the buffer capacities are small and the machine failure rates are large. Furthermore, Xia et al. (2016) propose an algorithm for non-homogeneous assembly systems with exponential machines, which shows higher approximation accuracy compared to that reported in Gershwin and Burman (2000). Despite the improved accuracy in throughput approximation, Xia et al. (2016) only study two medium-sized system structures (no more than 15 machines in each case) and the machines are selected to be highly efficient (efficiency over 90%). Moreover, the approximation error of buffer levels (i.e. work-in-process) can still be quite large in several cases studied (average error at 12.73%). In another extension work of the performance evaluation method, Wang et al. (2018) study non-homogeneous assembly/disassembly systems with multiple failure modes. However, the method is only tested for some small-scale systems (with three or five machines in total), and even for these small systems, the algorithm may still take up to several minutes to converge. Finally, assembly/disassembly systems with machines having no failures but generally distributed processing times are studied in Manitz (2008) and Manitz (2015).

Another commonly used method for assembly system performance metrics evaluation, referred to as *recursive aggregation* in the literature, has been applied in Kuo et al. (1997), Chiang et al. (2000), Li (2005), Jia et al. (2016), and Jia et al. (2019). The pioneering work in this direction can be traced to Kuo et al. (1997), which studies three-machine assembly systems (two component machines and one merge operation) under the Bernoulli reliability model and is later extended by Chiang et al. (2000) to assembly systems with one merge operation, two component lines, and a main line after the merge point. An arrow-based bottleneck identification method for such assembly system structure is proposed in Chiang et al. (2000). It is further justified by Ching, Meerkov, and Zhang (2008) that the method is effective for systems with non-Markovian machines as well. Assembly systems with Bernoulli machines under general structures are studied in Jia et al. (2016) and Jia et al. (2019), where the system's transient performance metrics evaluation, bottleneck identification, and

behavior under finite production runs are investigated. For assembly systems with non-Bernoulli machines, Li (2005) presents an approach, referred to as *overlapping decomposition*, for modeling and analysis of synchronous complex manufacturing systems consisting of assembly (disassembly), split, merge, parallel, and other complex operations with exponential machines. While the overlapping decomposition approach provides the idea that can be used to structurally decompose a multi-level assembly/disassembly systems, the algorithm/formulas reported in Li (2005) only apply to assembly operations with two component lines and it assumes that all machines have identical processing speed (i.e. homogeneous system).

1.3. Research gap and contribution of this work

Despite these valuable results, it is clear that the existing literature still does not have a computationally efficient algorithm that can accurately approximate a wider range of system performance metrics (throughput, work-in-process, probabilities of machine blockage and starvation, etc.) for multi-level, large-size assembly/disassembly systems with exponential machines. Thus, the goal of this research is to fill this gap. To accomplish this, preliminary work was conducted in Bai and Zhang (2021), which studies a class of simple assembly systems, where the main line of an assembly system has multiple merge operations, each connecting to only one component line and no sub-assemblies are involved. The study generalises the overlapping decomposition approach of Li (2005) and combines it with the new aggregation-based algorithm for performance evaluation of exponential serial lines developed in Bai et al. (2020). In the current paper, we further extend the study by including multi-tier sub-assemblies, disassembly operations, and allowing multiple component lines to connect to the same merge operation. This extension significantly enhances the applicability of the theoretical development as it covers a wide range of assembly/disassembly-based production system topologies, but at the same time calls for the development of new methods to quantify the behavior and interactions of more complicated system connections. To achieve this, two types of systems are considered: the first has a single merge (assembly) operation but multiple component lines and the second has a general structure and may consist of multiple merge/split operations and sub-assemblies. The former is used to derive the formulas of the analytical method, while the latter is used to illustrate how to generalise it to more complex cases with the former being a building block.

1.4. Organization of the paper

The remainder of this paper is organised as follows. A special case (multiple component lines merging at a single merge operation) is first presented in Section 2, including model assumptions, performance metrics definition, derivation of the calculation formulas of the algorithm, and numerical tests and results. Then, the extension to the general case (systems with multiple merge/split operations and sub-assemblies) is carried out in Section 3. An industrial case study is presented in Section 4 to demonstrate the practical utility of the theoretical methods developed. Finally, the conclusions and future work are summarised in Section 5.

2. Special case: single-merge-operation multiple-component-line assembly systems

In this section, we consider a special case: assembly systems with a single merge operation and multiple component lines (see Figure 1). Extension of the results to the general case will be carried out in Section 3.

2.1. Model assumptions

Consider the assembly system model shown in Figure 1 defined by the following assumptions:

- (i) The assembly system has a total of n component lines, CL_1, \dots, CL_n , and one main line.
- (ii) Component line CL_k has M_k machines (indicated by the circles in Figure 1), m_{k1}, \dots, m_{kM_k} and M_k buffers (indicated by the rectangles in Figure 1), b_{k1}, \dots, b_{kM_k} . The main line has M_0 machines and $M_0 - 1$ buffers. Machine m_{01} is the merge (i.e. assembly) operation.
- (iii) Each buffer is characterised by its finite capacity N_{ki} , $k = 1, \dots, n$, $i = 1, \dots, M_k$ and N_{0i} , $i = 1, \dots, M_0 - 1$, i.e. the maximal number of parts the buffer can hold.
- (iv) The system operates in continuous time. The machines are unreliable and subject to random failures. The uptime of machine m_{ki} , $k = 0, \dots, n$, $i = 1, \dots, M_k$, is an exponential random variable with parameter λ_{ki} (1/min). The downtime of machine m_{ki} is an exponential random variable with parameter μ_{ki} (1/min). Parameters λ_{ki} and μ_{ki} are referred to as the *breakdown rate* and *repair rate* of machine m_{ki} , respectively.
- (v) A machine is *starved* if it is up, its upstream buffer is empty, and its immediate upstream machine is either down or producing at a slower speed.

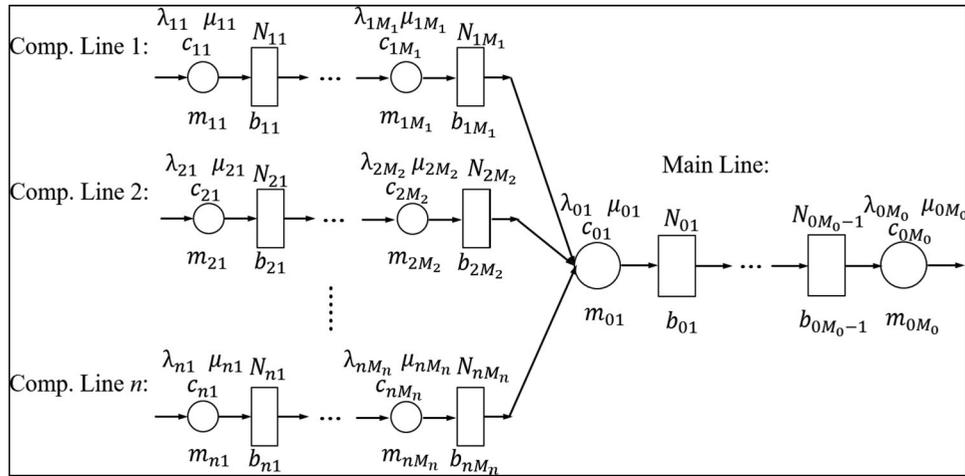


Figure 1. Single-merge-operation multiple-component-line assembly system.

Machine m_{k1} , $k = 1, \dots, n$, is never starved for raw material. The merge machine (m_{01}) is *starved* if it is up and any one of the upstream buffers is empty.

- (vi) A machine is *blocked* if it is up, its downstream buffer is full, and its immediate downstream machine is either down or producing at a slower speed. Machine m_{0M_0} is never blocked.
- (vii) Machine m_{ki} , $k = 0, \dots, n$, $i = 1, 2, \dots, M_k$, when up, can process jobs with speed up to c_{ki} (jobs/min). When being starved (blocked), it produces at the same speed as its immediate upstream (downstream) counterpart. For the merge operation m_{01} , it collects one unit of job from each of its preceding buffers ($b_{1M_1}, \dots, b_{nM_n}$) simultaneously, assembles all components into one job with speed up to c_{01} (jobs/min), and transfer the assembled job to buffer b_{01} . Note that the collection of components does not start until all components are available in the feeding buffers.

Clearly, the efficiency of machine m_{ki} can be calculated as:

$$e_{ki} = \frac{\mu_{ki}}{\lambda_{ki} + \mu_{ki}} = \frac{T_{up,ki}}{T_{up,ki} + T_{down,ki}}, \quad (1)$$

where $T_{up,ki} = 1/\lambda_{ki}$ and $T_{down,ki} = 1/\mu_{ki}$ are the average up- and downtimes, respectively. The stand-alone throughput of m_{ki} is, thus, $c_{ki}e_{ki}$ (jobs/min). In addition, we assume continuous job flows, time-dependent failures, and blocked-before-service conventions, to be consistent with the assumptions of our prior work in Bai et al. (2020). These conventions are also commonly used in production systems research and several industrial case studies using this type of production system model have been reported in the literature (see Li and

Meerkov 2008; Li et al. 2009; Li 2013; Colledani, Ratti, and Senanayake 2015).

2.2. Performance metrics considered and problem addressed

This paper considers the following steady state performance metrics of the production systems defined by assumptions (i)–(vii):

- *Throughput (TP)*: the expected rate of production by the last machine m_{0M_0} in the system during steady state.
- *Work-in-process (WIP_{ki})*: the expected number of jobs in buffer b_{ki} during steady state.
- *Machine starvation (ST_{ki})*: the probability that machine m_{ki} is starved by its immediate upstream buffer b_{ki-1} , $k = 0, 1, \dots, n$, $i = 2, \dots, M_k$. Note that for the merge operation m_{01} , n different starvation probabilities, $ST_{01,j}$, $j = 1, \dots, n$, should be considered, each corresponding to the starvation due to buffer b_{jM_j} .
- *Machine blockage (BL_{ki})*: the probability that machine m_{ki} is blocked by its immediate downstream buffer b_{ki} , $k = 0, \dots, n$, $i = 1, \dots, M_k$.

Note that the steady state behavior reflects the system’s long-term average performance, which is a commonly studied topic in manufacturing systems research and practice (see Bai et al. 2020; Ching, Meerkov, and Zhang 2008). Note also that the assembly system defined by the above assumptions is characterised by a continuous-time, mixed-state Markov process. However, direct Markovian analysis of such systems is not tractable due to the exponentially growing state space and the complexity incurred by the mix of discrete (machine

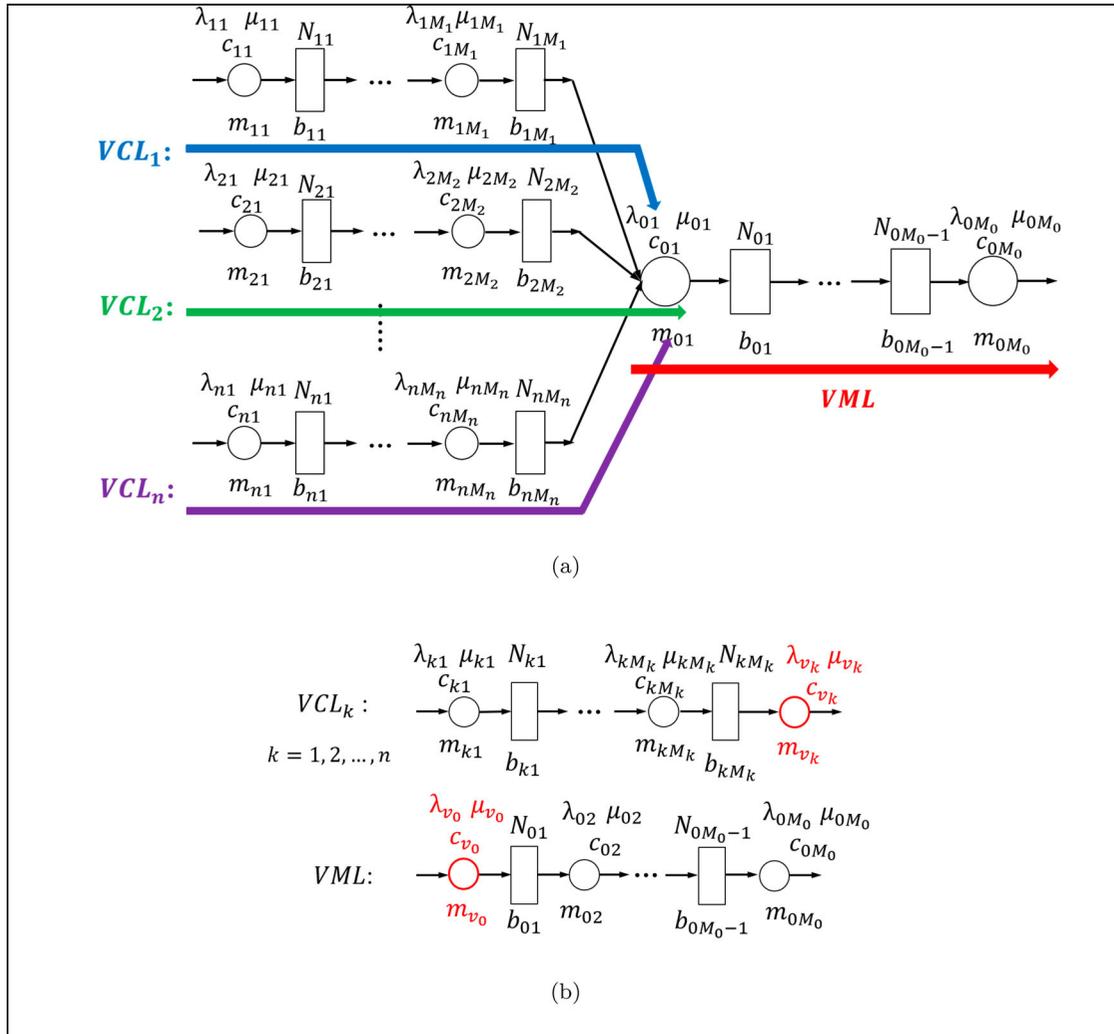


Figure 2. Decomposition of a single-merge-operation multi-component-line assembly system.

status $s_{ki} \in \{0 = \text{down}, 1 = \text{up}\}$) and continuous (buffer occupancy $h_{ki} \in [0, N_{ki}]$) states. On the other hand, simulations of the systems at hand may be time-consuming even for a moderate-sized system. Thus, in this paper, we aim at *developing computationally efficient and accurate analytical algorithms to calculate (approximate) the steady state performance metrics* defined above.

2.3. Approach

To analyze the assembly system at hand, we apply the *overlapping decomposition* approach developed in Li (2005) to decompose (approximate) the original assembly system in Figure 1 into $n + 1$ independent virtual serial lines (see Figure 2(a)) and assume that each virtual line is capable of representing the parts flow of the corresponding component line or the main line. Specifically, virtual component line VCL_k , $k = 1, \dots, n$, consists of all machines and buffers in the (original) Component Line k and a modified merge operation

(denoted as m_{v_k}), while the virtual main line VML consists of a modified version of the merge operation (denoted as m_{v_0}) and all its downstream machines and buffers in the (original) main line (see Figure 2(b)).

In each of these virtual lines, m_{v_k} (marked in red in Figure 2(b)) is introduced to incorporate the effects from the main line and *other* component lines (i.e. blockage due to buffer b_{01} and starvation when some of its preceding buffers b_{kM_k} 's are empty) into the parameters of the merge operation m_{01} . All other machines and buffers retain their original parameters. The goal of the decomposition is to construct the parts flows in the virtual lines so that they are similar to their counterparts in the original assembly system.

To accomplish that, we apply the aggregation algorithm developed in Bai et al. (2020) for *serial line* performance metric calculation to each virtual line constructed above. Using this method, each virtual line is represented as a group of virtual two-machine lines via the so-called *backward* (b) and *forward* (f) aggregations.

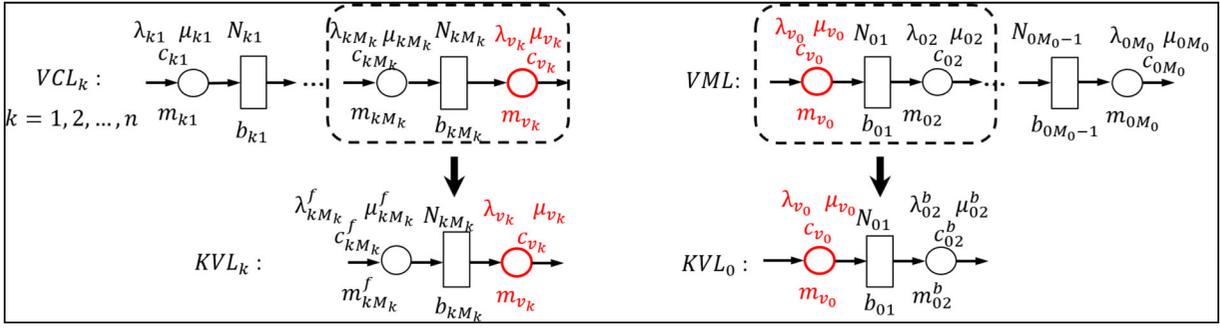


Figure 3. Key virtual two-machine lines

In particular, for the virtual two-machine line around buffer b_{kM_k} (see Figure 3) in VCL_k , virtual machine $m_{kM_k}^f$ represents (approximates) the aggregated part producing behavior of all machines and buffers upstream of b_{kM_k} . Based on this virtual two-machine line (referred to as *key virtual two-machine line k* or KVL_k) we can calculate the probabilities that the virtual merge operation m_{v_k} is starved (either partially or completely). These probabilities, denoted as $P_{uu,0}$ and $P_{du,0}$, where $u = up$ and $d = down$ represent the state of the two machines involved and 0 indicates that the buffer is empty in this state, will be used to calculate the parameters of the virtual merge operations in other virtual lines. Similarly, the virtual two-machine line around buffer b_{01} in VML (referred to as *key virtual two-machine line 0* or KVL_0) can be used to calculate the probabilities that m_{v_0} is blocked (denoted as $P_{ud,N}$ and $P_{uu,N}$, where N indicates that the buffer is full in this state). This coupling nature among the virtual lines enables a recursive procedure to continuously update the parameters of m_{v_k} 's for all virtual lines iteratively.

Finally, the performance metrics of the original assembly system can be obtained from the corresponding machines/buffers in the virtual lines. Mathematical formulas and detailed numerical implementation of the approach described above are given next.

2.4. Formulas to calculate parameters of the virtual merge operations

As mentioned above, the main components of the approach are the aggregation-based procedure for serial lines and the computation of the parameters of the virtual merge operation m_{v_k} 's. The former is based on our prior work (Bai et al. 2020). This subsection focuses on deriving the formulas for the latter.

To achieve this, first consider the key virtual two-machine lines shown in Figure 3 and introduce

- $P_{uu,0}^{(k)}$: the probability that b_{kM_k} in KVL_k is empty, and both $m_{kM_k}^f$ and m_{v_k} are up, $k = 1, \dots, n$;
- $P_{du,0}^{(k)}$: the probability that b_{kM_k} in KVL_k is empty, $m_{kM_k}^f$ is down and m_{v_k} is up, $k = 1, \dots, n$;
- $P_{uu,N}^{(0)}$: the probability that b_{01} in KVL_0 is full, both m_{v_0} and m_{02}^b are up;
- $P_{ud,N}^{(0)}$: the probability that b_{01} in KVL_0 is full, m_{v_0} is up and m_{02}^b is down.

Note that the downstream (upstream) machine is *partially* starved (blocked) in the $uu, 0$ (uu, N) case when its processing speed is greater than the upstream (downstream) one. In these cases, the processing speed of the faster machine is reduced to that of the slower one. On the other hand, in the $du, 0$ and ud, N cases, the corresponding *up* machines are completely starved and blocked, respectively, and no part processing is carried out. Next, we will use these notations to derive formulas to calculate the parameters of the virtual merge operations.

2.4.1. Breakdown and repair rates of m_{v_k}

Consider virtual merge operation m_{v_k} in VCL_k . To incorporate the effects from other parts of the assembly system (i.e. $VCL_i, i = 1, \dots, n, i \neq k$ and VML) into the parameters of m_{v_k} , note that from the perspective of CL_k , merge operation m_{01} may be operating in one of the following cases:

- Cases $C_1 - C_5$: Up
 - Case C_1 : Up, not blocked by buffer b_{01} nor starved by any upstream buffer in the other component lines, $b_{iM_i}, i = 1, \dots, n, i \neq k$: In this case, m_{01} may process parts with its full speed c_{01} in VCL_k (as m_{v_k});
 - Case C_2 : Up, no blockage from buffer b_{01} , but partially starved by some of its upstream buffers b_{iM_i}

in the other component lines: In this case, m_{01} processes parts in VCL_k (as m_{v_k}) at a reduced speed determined by the component lines that cause partial starvation;

- Case \mathcal{C}_3 : Up, no starvation from any of its upstream buffers b_{iM_i} in the other component lines, but partially blocked by buffer b_{01} : In this case, m_{01} processes parts in VCL_k (as m_{v_k}) at a reduced speed determined by its downstream operations in the main line;
- Case \mathcal{C}_4 : Up, partially blocked by buffer b_{01} , and partially starved by some of its upstream buffers b_{iM_i} in the other component lines: In this case, m_{01} processes parts in VCL_k (as m_{v_k}) at a reduced speed determined by the component lines that cause partial starvation as well as its downstream operations in the main line;
- Case \mathcal{C}_5 : Up and either completely blocked by buffer b_{01} or completely starved by at least one of its upstream buffers b_{iM_i} in the other component lines: In this case, no production by m_{01} is possible;
- Case \mathcal{C}_6 : Down: no production is possible.

Similarly, from the perspective of the main line, merge operation m_{01} may be operating in one of the following cases:

- Cases $\mathcal{M}_1 - \mathcal{M}_3$ Up
 - Case \mathcal{M}_1 : Up, not starved by any upstream buffer b_{iM_i} , $i = 1, \dots, n$: In this case, m_{01} may process parts with its full speed c_{01} in VML (as m_{v_0});
 - Case \mathcal{M}_2 : Up, partially starved by some of its upstream buffers b_{iM_i} : In this case, m_{01} processes parts in VML (as m_{v_0}) at a reduced speed determined by the component lines that cause partial starvation;
 - Case \mathcal{M}_3 : Up and completely starved by at least one of its upstream buffers b_{iM_i} : In this case, no production by m_{01} is possible;
- Case \mathcal{M}_4 : Down: no production is possible.

From the point of view of machine m_{v_k} in VCL_k , both Cases \mathcal{C}_5 and \mathcal{C}_6 can be viewed as downtime since no part-processing can be conducted in these cases. To account for the additional downtime introduced in Case \mathcal{C}_5 , note that, the fraction of uptime that m_{01} is completely starved by buffer b_{iM_i} in CL_i can be approximated from KVL_i as $\frac{P_{du,0}^{(i)}}{e_{v_i}}$. Similarly, the fraction of uptime that merge operation m_{01} is completely blocked by buffer b_{01} in the main line can be approximated from KVL_0 as $\frac{P_{ud,N}^{(0)}}{e_{v_0}}$. Therefore,

we define

$$P_{op}^{(k)} = \left(1 - \frac{P_{ud,N}^{(0)}}{e_{v_0}}\right) \prod_{i=1, i \neq k}^n \left(1 - \frac{P_{du,0}^{(i)}}{e_{v_i}}\right) \quad (2)$$

and similarly, to account for the additional downtime of m_{v_0} caused by Case \mathcal{M}_3 , define

$$P_{op}^{(0)} = \prod_{i=1}^n \left(1 - \frac{P_{du,0}^{(i)}}{e_{v_i}}\right). \quad (3)$$

Here, subscript *op* stands for ‘operational’. Clearly, variable $P_{op}^{(k)}$ can be viewed as an estimated probability that the merge operation in VCL_k or VML is *not* stopped due to starvation or blockage from other virtual lines, and thus, is *operational*. To incorporate these effects into the machine parameters, we define the efficiency and repair rate of virtual merge operation m_{v_k} by reducing the original parameters of m_{01} by a factor of $P_{op}^{(k)}$:

$$e_{v_k} = e_{01} P_{op}^{(k)}, \quad \mu_{v_k} = \mu_{01} P_{op}^{(k)}, \quad k = 0, \dots, n, \quad (4)$$

which immediately leads to

$$\lambda_{v_k} = \lambda_{01} + \mu_{01} (1 - P_{op}^{(k)}). \quad (5)$$

In other words, the stoppages due to complete starvation or blockage by other virtual lines are now accounted for by reducing the efficiency and repair rate and increasing the breakdown rate of the merge operation from its original parameters.

2.4.2. Processing speed of m_{v_k}

Note that, while virtual merge operation m_{v_k} can be viewed as being *up* in Cases $\mathcal{C}_1 - \mathcal{C}_4$, its actual processing speed depends on the states of other component lines and the the main line connected to CL_k via m_{01} . In this subsection, we derive (estimate) the probability of each case involved, the corresponding processing speed of the merge operation for each case, and a normalised average processing speed that combines all four cases.

Case \mathcal{C}_1 : *No external starvation/blockage on m_{v_k}* When the merge operation is neither starved nor blocked in any of the other virtual lines, it may operate in VCL_k with its original full processing speed c_{01} . Similar to Equations (2) and (3), the conditional probability of this scenario (given that the merge operation is up) can be quantified as

$$P_{fs}^{(k)} = \left(1 - \frac{P_{ud,N}^{(0)} + P_{uu,N}^{(0)}}{e_{v_0}}\right) \times \prod_{i=1, i \neq k}^n \left(1 - \frac{P_{uu,0}^{(i)} + P_{du,0}^{(i)}}{e_{v_i}}\right), \quad k = 1, 2, \dots, n, \quad (6)$$

where subscript *fs* stands for ‘full speed’.

Case C₂: No external blockage and only (partial) starvation on m_{v_k} Consider the case where the merge operation is not blocked by the main line but only (potentially) partially starved by q component lines other than CL_k , while the remaining $n-q-1$ component lines present no starvation. Note that there are $\binom{n-1}{q}$ different possible combinations of the starvation-causing component lines. Thus, we define $\binom{n-1}{q} \times q$ matrix \mathbf{I}_q such that its i -th row contains the indices of the q virtual lines that are (potentially) partially starving the merge operation under this combination. Similarly, define $\binom{n-1}{q} \times (n-q-1)$ matrix $\bar{\mathbf{I}}_q$ such that each row of it contains the indices of the remaining $n-q-1$ component lines that are not starving the merge operation. Then, for the i -th combination of all q -machine partial starvation case, its conditional probability can be quantified as

$$P_{ps,i,q}^{nb} = \left(1 - \frac{P_{ud,N}^{(0)} + P_{uu,N}^{(0)}}{e_{v_0}}\right) \prod_{j=1}^q \frac{P_{uu,0}^{(\mathbf{I}_{q,i,j})}}{e_{v_{\mathbf{I}_{q,i,j}}}} \cdot \prod_{j=1}^{n-q-1} \left(1 - \frac{P_{uu,0}^{(\bar{\mathbf{I}}_{q,i,j})} + P_{du,0}^{(\bar{\mathbf{I}}_{q,i,j})}}{e_{v_{\bar{\mathbf{I}}_{q,i,j}}}}\right), \quad (7)$$

$$i = 1, \dots, \binom{n-1}{q},$$

and the virtual merge machine's processing speed in this scenario is the smallest among all machines involved, i.e.

$$c_{ps,i,q}^{nb} = \min(c_{\mathbf{I}_{q,i,1}, M_{\mathbf{I}_{q,i,1}}}^f, \dots, c_{\mathbf{I}_{q,i,q}, M_{\mathbf{I}_{q,i,q}}}^f, c_{01}), \quad (8)$$

where $\mathbf{I}_{q,i,j}$ and $\bar{\mathbf{I}}_{q,i,j}$ denote the element at the i -th row and j -th column in matrices \mathbf{I}_q and $\bar{\mathbf{I}}_q$, respectively; $c_{\mathbf{I}_{q,i,j}, M_{\mathbf{I}_{q,i,j}}}^f$ is the processing speed of the upstream machine of $KVL_{\mathbf{I}_{q,i,j}}$ (see Figure 3); and subscript/superscript ps and nb stand for 'partial starvation' and 'no blockage', respectively.

Case C₃: No external starvation and only (partial) blockage on m_{v_k} Assume that there is no starvation from all other component lines, and the merge operation is only partially blocked by b_{01} in the main line. Then, the conditional probability for this scenario can be expressed as

$$P_{ns,k}^{bl} = \frac{P_{uu,N}^{(0)}}{e_{v_0}} \cdot \prod_{i=1, i \neq k}^n \left(1 - \frac{P_{uu,0}^{(i)} + P_{du,0}^{(i)}}{e_{v_i}}\right), \quad (9)$$

and the virtual merge machine m_{v_k} 's processing speed in this scenario can be calculated as

$$c_{ns,k}^{bl} = \min(c_{02}^b, c_{01}), \quad (10)$$

where c_{02}^b is the processing speed of the downstream machine in KVL_0 (see Figure 3) and subscript/superscript

ns and bl stand for 'no starvation' and 'blockage', respectively.

Case C₄: Simultaneous external blockage and starvation on m_{v_k} When, in addition to being starved in q component lines, the merge operation is also blocked by the main line, the conditional probability for this case can be calculated as

$$P_{ps,i,q}^{bl} = \frac{P_{uu,N}^{(0)}}{e_{v_0}} \cdot \prod_{j=1}^q \frac{P_{uu,0}^{(\mathbf{I}_{q,i,j})}}{e_{v_{\mathbf{I}_{q,i,j}}}} \cdot \prod_{j=1}^{n-q-1} \left(1 - \frac{P_{uu,0}^{(\bar{\mathbf{I}}_{q,i,j})} + P_{du,0}^{(\bar{\mathbf{I}}_{q,i,j})}}{e_{v_{\bar{\mathbf{I}}_{q,i,j}}}}\right), \quad (11)$$

and the merge machine's processing speed in this case is

$$c_{ps,i,q}^{bl} = \min(c_{\mathbf{I}_{q,i,1}, M_{\mathbf{I}_{q,i,1}}}^f, \dots, c_{\mathbf{I}_{q,i,q}, M_{\mathbf{I}_{q,i,q}}}^f, c_{01}, c_{02}^b), \quad (12)$$

where subscript/superscript ps and bl stand for 'partial starvation' and 'blockage', respectively.

Finally, combining the above four cases, the processing speed of virtual merge machine m_{v_k} in VCL_k is defined as their weighed average:

$$c_{v_k} = \frac{1}{P_{op}^{(k)}} \cdot \left(c_{01} P_{fs}^{(k)} + \sum_{q=1}^{n-1} \sum_{i=1}^{\binom{n-1}{q}} c_{ps,i,q}^{nb} P_{ps,i,q}^{nb} + c_{ns,k}^{bl} P_{ns,k}^{bl} + \sum_{q=1}^{n-1} \sum_{i=1}^{\binom{n-1}{q}} c_{ps,i,q}^{bl} P_{ps,i,q}^{bl} \right), \quad k = 1, 2, \dots, n. \quad (13)$$

Similarly, the processing speed of m_{v_0} can be calculated based on the production scenarios of Cases \mathcal{M}_1 and \mathcal{M}_2 as follows:

$$c_{v_0} = \frac{1}{P_{op}^{(0)}} \cdot \left(c_{01} P_{fs}^{(0)} + \sum_{q=1}^n \sum_{i=1}^{\binom{n}{q}} c_{ps,i,q} P_{ps,i,q} \right), \quad (14)$$

where

$$P_{fs}^{(0)} = \prod_{i=1}^n \left(1 - \frac{P_{uu,0}^{(i)} + P_{du,0}^{(i)}}{e_{v_i}}\right), \quad (\text{Case } \mathcal{M}_1) \quad (15)$$

$$P_{ps,i,q} = \prod_{j=1}^q \frac{P_{uu,0}^{(\mathbf{I}_{q,i,j})}}{e_{v_{\mathbf{I}_{q,i,j}}}} \cdot \prod_{j=1}^{n-q} \left(1 - \frac{P_{uu,0}^{(\bar{\mathbf{I}}_{q,i,j})} + P_{du,0}^{(\bar{\mathbf{I}}_{q,i,j})}}{e_{v_{\bar{\mathbf{I}}_{q,i,j}}}}\right), \quad (\text{Case } \mathcal{M}_2) \quad (16)$$

$$c_{ps,i,q} = \min(c_{\mathbf{I}_{q,i,1}, M_{\mathbf{I}_{q,i,1}}}^f, \dots, c_{\mathbf{I}_{q,i,q}, M_{\mathbf{I}_{q,i,q}}}^f, c_{01}). \quad (17)$$

It should be noted that when the system is synchronous (i.e. when all c_{ki} 's are identical), partial blockage and starvation do not take place and, thus, the virtual machine's

processing speed (including m_{v_k} 's) remains at their original values during the aggregation procedure. If we further limit the number of component lines to $n = 2$, then the formulas (2)–(5) become identical to those presented in Li (2005). In other words, the assembly system case studied in Li (2005) is a special case (synchronous and $n = 2$) of the one discussed in this section.

2.5. Numerical implementation

Based on the descriptions above, the following recursive algorithm (referred to as *Recursive Decomposition/Aggregation Algorithm I, RDAA I*) is proposed for performance metrics calculation in single-merge-operation multiple-component-line assembly systems. The idea is to circulate among all virtual lines, one at a time, to update the parameters of the merge operations in the virtual lines based on Equations (2)–(17) and repeat this process iteratively until a stopping criterion is met. Note that in the pseudo code of the algorithm, s is the iteration counter and $\widetilde{TP}_k(s)$ is the throughput of the k -th virtual line calculated using the aggregation procedure of Bai et al. (2020).

Unfortunately, the proposed aggregation algorithm, RDAA I, does not have an analytical proof of convergence at this point. As an alternative, we investigate its convergence through numerical experiments. Specifically, we create a C++ program to implement the algorithm and generate a total of 4,000,000 assembly systems by randomly selecting the system parameters from:

$$n \in \{2, 3, 4, 5\}, \quad M_k \in \{2, 3, 4, 5\}, \quad k = 0, \dots, n. \quad (18)$$

For each machine m_{ki} , the parameters are randomly selected from the following ranges and independently from other machines:

$$\begin{aligned} T_{down,ki} &\in [5, 15], & e_{ki} &\in [0.75, 0.95], \\ c_{ki} &\in [0.8, 1.2], & k = 0, \dots, n, & \quad i = 1, \dots, M_k. \end{aligned} \quad (19)$$

The capacities of the buffers are determined based on

$$N_{ki} = \lceil K_{ki} \cdot \min(T_{down,ki}, T_{down,ki+1}) \rceil, \quad (20)$$

where each coefficient K_{ki} , $k = 0, \dots, n$, $i = 1, \dots, M_k$, is randomly selected from interval $[1, 3]$. Note that these parameter ranges are selected to reflect typical observations in manufacturing practice (see Li and Meerkov 2008; Li 2005). Under the above parameter ranges, the maximal difference in machine processing speeds is 50%. In other words, the production systems studied can be viewed as balanced or *moderately* unbalanced. For buffer capacity, the parameter K_{ki} represents the number of average downtimes that a buffer can accommodate. In the numerical studies, K_{ki} 's are selected from interval $[1, 3]$, which means that the buffering provided is in the range of small to medium levels (see Li and Meerkov 2008). Very small or large buffers are not very common in manufacturing practices.

The terminal condition is set as $\Delta(s) < 10^{-5}$. For all 4,000,000 assembly systems, the algorithm reaches this terminal condition without exception. Therefore, we claim that RDAA I is convergent (numerically) and

Algorithm 1 Recursive Decomposition/Aggregation Algorithm I (RDAA I)

Initialization: Set

$$s = 0, \Delta(s) = 1;$$

$$\lambda_{v_k} = \lambda_{01}, \mu_{v_k} = \mu_{01}, c_{v_k} = c_{01}, k = 1, \dots, n;$$

$$P_{uu,N}^{(0)} = P_{ud,N}^{(0)} = P_{uu,0}^{(k)} = P_{du,0}^{(k)} = 0, k = 1, \dots, n;$$

$$\widetilde{TP}(s) = [\widetilde{TP}_0(s), \dots, \widetilde{TP}_n(s)] = [0, \dots, 0].$$

while $\Delta(s) \geq \epsilon$ **do**

$$s = s + 1;$$

1. Update parameters λ_{v_0} , μ_{v_0} and c_{v_0} based on (3)–(5), (14)–(17);

2. Apply the aggregation algorithm developed in Bai et al. (2020) to VML to calculate its throughput $\widetilde{TP}_0(s)$, as well as $P_{uu,N}^{(0)}$ and $P_{ud,N}^{(0)}$ of KVL_0 ;

for $k = 1; k \leq n$ **do**

1. Update parameters λ_{v_k} , μ_{v_k} , and c_{v_k} based on (2), (4)–(13);

2. Apply the aggregation algorithm developed in Bai et al. (2020) to VCL_k to calculate its throughput $\widetilde{TP}_k(s)$, as well as $P_{uu,0}^{(k)}$ and $P_{du,0}^{(k)}$ of KVL_k ;

end for

$$\Delta(s) = \|\widetilde{TP}(s) - \widetilde{TP}(s-1)\|;$$

end while

introduce the following limits:

$$\begin{aligned} \lim_{s \rightarrow \infty} \lambda_{v_k}(s) &= \lambda_{v_k}, & \lim_{s \rightarrow \infty} \mu_{v_k}(s) &= \mu_{v_k}, \\ \lim_{s \rightarrow \infty} c_{v_k}(s) &= c_{v_k}, & k &= 0, 1, \dots, n. \end{aligned} \quad (21)$$

Moreover, it is observed that RDAA I requires, on average, about 10 iterations to converge for the cases studied above. The average CPU computational time on a PC with Intel®Core™ i7-9700H 3.0GHz processor and 16GB RAM is less than 1.7 ms, which is almost negligible compared to discrete-event simulations.

2.6. Performance metrics calculation and accuracy

2.6.1. Performance metrics calculation

Using the limiting values (21) resulted from RDAA I and the aggregation algorithm of Bai et al. (2020), we can calculate the performance metrics of each virtual line (i.e. VCL_i 's and VML), denoted as \widehat{TP}_k , \widehat{WIP}_{ki} , \widehat{ST}_{ki} , and \widehat{BL}_{ki} , $k = 0, 1, \dots, n$, $i = 1, \dots, M_k$. Then, we propose to calculate the performance measures of the original assembly system as follows:

- *System throughput*: $\widehat{TP} = \text{mean}(\widehat{TP})$.
- *Work in process*:
 - Component lines: $\widehat{WIP}_{ki} = \widehat{WIP}_{ki}$, $k = 1, \dots, n$, $i = 1, \dots, M_k$;
 - Main line: $\widehat{WIP}_{0i} = \widehat{WIP}_{0i}$, $i = 1, \dots, M_0 - 1$.
- *Starvation*:
 - Component lines: $\widehat{ST}_{ki} = \widehat{ST}_{ki}$, $k = 1, \dots, n$, $i = 1, \dots, M_k$;
 - Main line: $\widehat{ST}_{0i} = \widehat{ST}_{0i}$, $i = 2, \dots, M_0$; $\widehat{ST}_{01,k} = \widehat{ST}_{k,M_k+1}$, $k = 1, \dots, n$.
- *Blockage*:
 - Component lines: $\widehat{BL}_{ki} = \widehat{BL}_{ki}$, $k = 1, \dots, n$, $i = 1, \dots, M_k$;
 - Main line: $\widehat{BL}_{0i} = \widehat{BL}_{0i}$, $i = 1, \dots, M_0 - 1$.

2.6.2. Accuracy of system performance metric approximation

To study the accuracy of the proposed performance metric approximations above, 40,000 single-merge-operation multiple-component-line systems (10,000 systems for each $n \in \{2, 3, 4, 5\}$) are randomly generated according to the parameter range mentioned in (18)–(20). In addition, a C++ program is created for the simulation of the assembly system models considered in this paper. This program is similar to the computational engines used in typical discrete-event simulation software packages. For each line constructed above, the simulation program runs 10 replications, with a warm-up time of 40,000 min and results collection time of 400,000 min (both in terms of simulation clock time) in each replication. Then, the

average values of the 10 simulation replications are used to estimate the true values of the performance metrics. The accuracy of the performance metric approximations is evaluated based on the following relative errors:

$$\begin{aligned} \epsilon_{TP} &= \frac{|\widehat{TP} - TP^{sim}|}{TP^{sim}} \cdot 100\%, \\ \epsilon_{WIP} &= \frac{1}{\sum_{j=0}^n M_j - 1} \left\{ \sum_{k=1}^n \sum_{i=1}^{M_k} \frac{|\widehat{WIP}_{ki} - WIP_{ki}^{sim}|}{N_{ki}} + \sum_{i=0}^{M_0-1} \frac{|\widehat{WIP}_{0i} - WIP_{0i}^{sim}|}{N_{0i}} \right\} \cdot 100\%, \\ \epsilon_{ST} &= \frac{1}{\sum_{j=0}^n M_j - 1} \left\{ \sum_{k=1}^n \sum_{i=1}^{M_k} |\widehat{ST}_{ki} - ST_{ki}^{sim}| + \sum_{i=2}^{M_0} |\widehat{ST}_{0i} - ST_{0i}^{sim}| + \sum_{i=1}^n |\widehat{ST}_{01,i} - ST_{01,i}^{sim}| \right\}, \\ \epsilon_{BL} &= \frac{1}{\sum_{j=0}^n M_j - 1} \sum_{k=0}^n \sum_{i=1}^{M_k} |\widehat{BL}_{ki} - BL_{ki}^{sim}|, \end{aligned} \quad (22)$$

where superscript *sim* indicates the system performance metrics obtained by simulation.

The results are summarised in Table 1(a-b). For $n = 2$, we also compare the performance of the proposed algorithm with the one developed in Li and Meerkov (2008) (referred to as *c*-Aggregation). As one can see, the proposed algorithm dramatically outperforms the one in Li and Meerkov (2008) and maintains a relatively low error even when the total number of machines in the system increases to over 21 (when $n = 5$). The *TP* approximation accuracy is similar to those reported in Xia et al. (2016) but has significantly improved *WIP* approximation accuracy. In addition, while the computational time increases as the system become more complex, the algorithm maintains relatively efficient, requiring only about 1.7 ms for systems with $n = 5$ component lines (about 21 machines in total on average). As a comparison, we also provide the average computational time needed when calculating the same systems' performance using simulation in Table 1(c). The configuration of the simulation (i.e. warm-up time, results collection time, number of replications) is selected so that the simulation outputs a similar level of precision compared to the accuracy of RDAA I. Clearly, RDAA I requires orders of magnitude shorter computational time. This high computational efficiency of the proposed algorithm is critical in tasks such as parameter optimization and design, where hundreds or even thousands of alternative parameter combinations must be evaluated and tested.

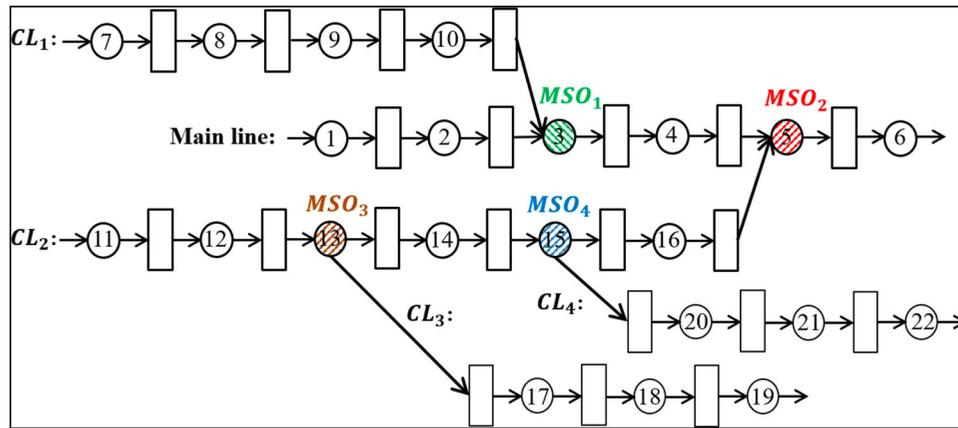


Figure 4. Example of a general A/D system.

Table 1. Accuracy and computational efficiency of RDAA I.

(a) Approximation error comparison for $n = 2$					
	ϵ_{TP}	ϵ_{WIP}	ϵ_{ST}	ϵ_{BL}	
RDAA I	2.29%	2.29%	0.0114	0.0117	
<i>c</i> -Aggregation (Li and Meerkov 2008)	13.59%	9.85%	0.0404	0.0527	
(b) Approximation error of RDAA I for $n \in \{3, 4, 5\}$					
	ϵ_{TP}	ϵ_{WIP}	ϵ_{ST}	ϵ_{BL}	Ave. $\sum M_k$
$n = 3$	2.22%	2.20%	0.0124	0.0121	14
$n = 4$	1.98%	2.08%	0.0132	0.0121	18
$n = 5$	1.82%	1.98%	0.0141	0.0127	21
(c) Computational time comparison with simulation					
	$n = 2$	$n = 3$	$n = 4$	$n = 5$	
RDAA I (ms)	0.10	0.35	0.74	1.7	
Simulation (ms)	90	107	130	150	

3. General case

In this section, we extend the analysis above to assembly/disassembly (A/D) systems with general structures, which may have multiple merge/split operations (MSOs) and sub-assemblies and dis-assemblies. Such a system consists of a main line, several first-level component lines (those directly joining or branching out from the main line), second-level component lines (those joining a first-level component line before merging into the main line or /branching out from a first-level component line), ..., up to several r -th level component lines (those joining/branching out from an $(r - 1)$ -th level component line). An example multi-level assembly/disassembly system with $r = 2$ is shown in Figure 4, which has 4 merge/split operations ($MSO_i, i = 1, \dots, 4$), 2 first-level assembly component lines (CL_1 and CL_2) and 2 second-level disassembly component lines (CL_3 and CL_4). For the merge/split operations, two of them are in the main line (MSO_1 and MSO_2) and two in first-level component lines (MSO_3 and MSO_4). The system operates following similar assumptions to those given in (i)–(vii).

3.1. System decomposition

Using the overlapping decomposition method, we can again decompose a complex assembly system into a number of serial lines. Specifically, first start with the first machine in a component line or the first machine of the main line and continue adding downstream machines and buffers along the direction of the parts flow and end at the first merge/split operation reached. Then, start from each merge/split operation and continue adding downstream machines and buffers until the next merge/split operation or the last machine in the main line or a disassembly component line is reached. This decomposition is illustrated in Figure 5(a) for the example assembly/disassembly system above, which leads to 9 virtual serial lines. Note that the main line or a component line may be sectioned into several virtual lines, depending on the number of merge operations in that line. For instance, the main line in this example ends up with 3 virtual lines ($VL_{0,1}, \dots, VL_{0,3}$), while component line CL_2 is decomposed into 3 virtual lines ($VL_{2,1}, VL_{2,2}, VL_{2,3}$).

Next, we group these virtual lines into a number of assembly/disassembly units (ADUs) based on their *overlapped* operation, i.e. the virtual merge operation. Specifically, for each merge/split operation, we group all virtual lines containing this merge/split operation together to form an ADU. Therefore, the number of ADUs is equal to the number of merge/split operations in the system. Note that a merge/split operation may appear in multiple ADUs but only acts as the ‘merge/split operation’ in exactly one ADU. This leads to 4 ADUs for the example assembly/disassembly system considered above (see Figure 5(a)).

Note that each ADU can be viewed as a single-merge/split-operation multiple-component-line assembly/disassembly system (see Figure 5(b)). Note also that some virtual lines may appear in more than one ADUs.

For instance, $VL_{0,2}$ acts as the ‘main line’ in ADU_1 and as a component line in ADU_2 . To distinguish them, they are denoted as $VL_{0,2}^{(1)}$ and $VL_{0,2}^{(2)}$, respectively, where the superscript denotes the ADU that the virtual line belongs to. Similarly, the virtual merge/split operations involved in VL_j are denoted as $MO_i^{(j)}$ (see Figure 5(b)). Clearly, the parameters of virtual merge/split operation $MSO_1^{(0,2)}$ in ADU_2 should account for the starvation of MSO_1 from its upstream component lines (i.e. $VL_{0,1}$ and VL_1), which can be obtained (approximated) from ADU_1 . Conversely, the parameters of $MSO_2^{(0,2)}$ in ADU_1 should account for the blockage of MSO_2 by its downstream buffer in $VL_{0,3}$, and the starvation from its upstream buffers in $VL_{2,3}$, which can be calculated from ADU_2 . The coupling relationships among the other ADUs can be identified similarly. Thus, virtual line $VL_{0,2}$, as an individual, isolated serial line, can be represented as in Figure 5(c), with the first and last machines being virtual merge/split operations $MSO_1^{(0,2)}$ and $MSO_2^{(0,2)}$, respectively. Then, applying the aggregation procedure of Bai et al. (2020) to this virtual serial line, we can obtain the probabilities $P_{uu,N}$ and $P_{ud,N}$ for the buffer(s) immediately after $MSO_1^{(0,2)}$ as well as the probabilities $P_{uu,0}$ and $P_{du,0}$ for the buffer(s)

immediately before $MSO_2^{(0,2)}$. These probabilities can be used to quantify the blockage of MSO_1 and the starvation of MSO_2 caused in the main line. Hence, we can generalise the idea of RDAA I through the ADUs, and utilise the starvation/blockage-induced coupling relationships to calculate the parameters of the virtual merge operations using Equations (2)–(16). Finally, repeating this process anew iteratively, we can continuously update the parameters of all virtual merge/split operations in a recursive manner until convergence is observed.

3.2. Numerical implementation

Based on the system decomposition described above, we propose the following Recursive Decomposition/Aggregation Algorithm II (RDAA II) to calculate the performance metrics of general assembly systems.

Similar to Subsection 2.4.1, numerical experiments are used to verify the convergence of the algorithm. Specifically, a total of 6,000 assembly/disassembly systems are generated with 2,000 for each of $r = 1, 2, 3$, where r is the number of component line levels of an assembly/disassembly system. For each system, after r is selected, the number of component lines at the i -th

Algorithm 2 Recursive Decomposition/Aggregation Algorithm II (RDAA II)

Step 1: Decomposition

Decompose the assembly/disassembly system into L virtual serial lines and group them into K assembly/disassembly units: ADU_1, \dots, ADU_K .

Step 2: Aggregation

Initialization: Set

$$s = 0, \Delta(s) = 1;$$

$$\widetilde{TP}(s) = [\widetilde{TP}_1(s), \widetilde{TP}_2(s), \dots, \widetilde{TP}_L(s)] = [0, 0, \dots, 0];$$

$$P_{uu,0}^{(i)} = P_{du,0}^{(i)} = P_{uu,N}^{(i)} = P_{ud,N}^{(i)} = 0, i = 1, 2, \dots, L;$$

Parameters of virtual MSOs = Original parameters of the corresponding MSOs

while $\Delta(s) \geq \epsilon$ **do**

$$s = s + 1;$$

for $i = 1; i \leq L$ **do**

1. Consider virtual serial production line VL_i .

2. Identify the ADUs that VL_i belongs to and denote the corresponding lines in these ADUs as $VL_i^{(j)}$, where j represents the indices of the ADUs and $j \in \{1, 2, \dots, K\}$.

for all j where ADU_j contains VL_i **do**

→ Update the parameters of the virtual $MSO_j^{(i)}$ in VL_i based on the starvation/blockage coupling with other virtual lines in ADU_j , quantified by $P_{uu,0}, P_{du,0}, P_{uu,N}$ and $P_{ud,N}$;

end for

3. Apply the aggregation algorithm developed in Bai et al. (2020) to VL_i with updated parameters to calculate its throughput $\widetilde{TP}_i(s)$, as well as the starvation/blockage probabilities $P_{uu,0}^{(i)}, P_{du,0}^{(i)}, P_{uu,N}^{(i)}$ and $P_{ud,N}^{(i)}$ for the virtual MSOs involved;

end for

$$\Delta(s) = \|\widetilde{TP}(s) - \widetilde{TP}(s-1)\|;$$

end while

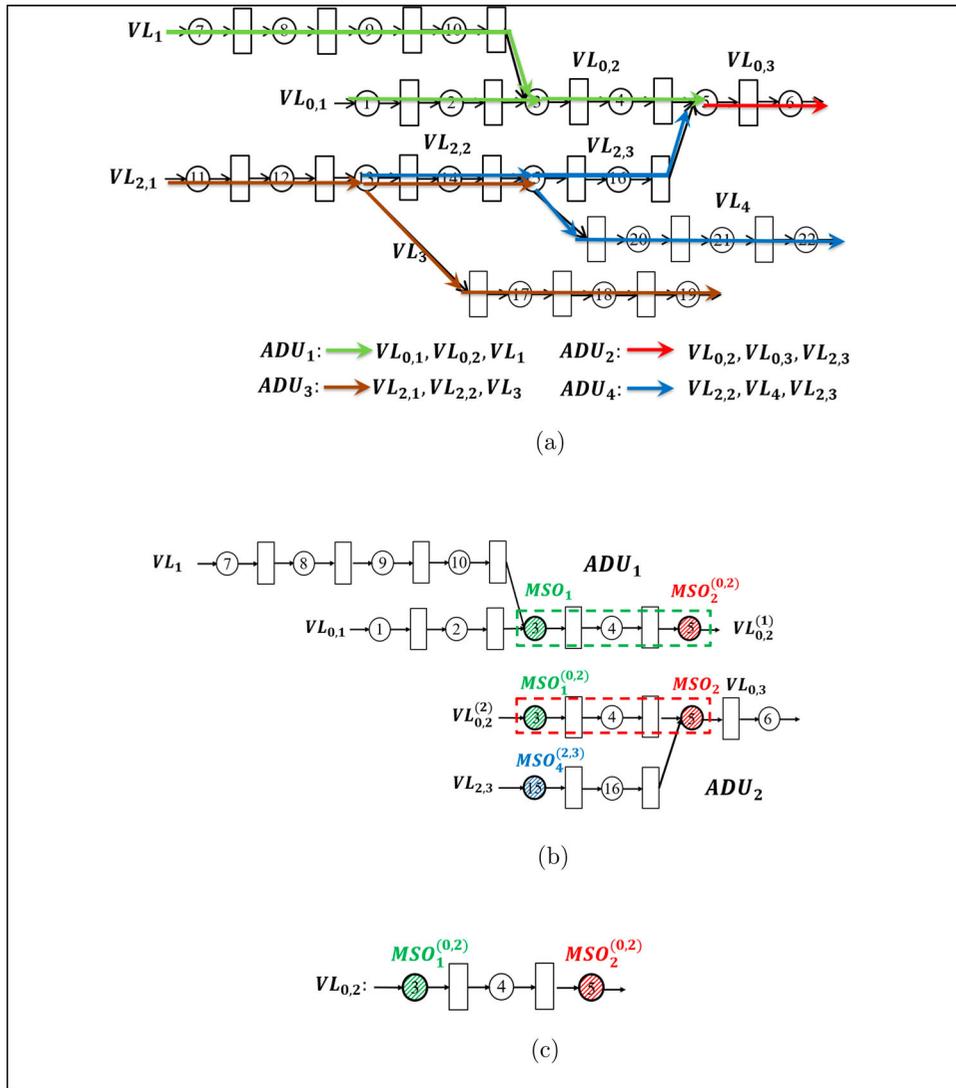


Figure 5. Illustration of the decomposition and analysis procedure of a general A/D system. (a) Example of decomposition of a general structured A/D system (b) Assembly/Disassembly Units with MSO₁ and MSO₂ being the merge/split operations (c) Virtual serial production line VL_{0,2} considered

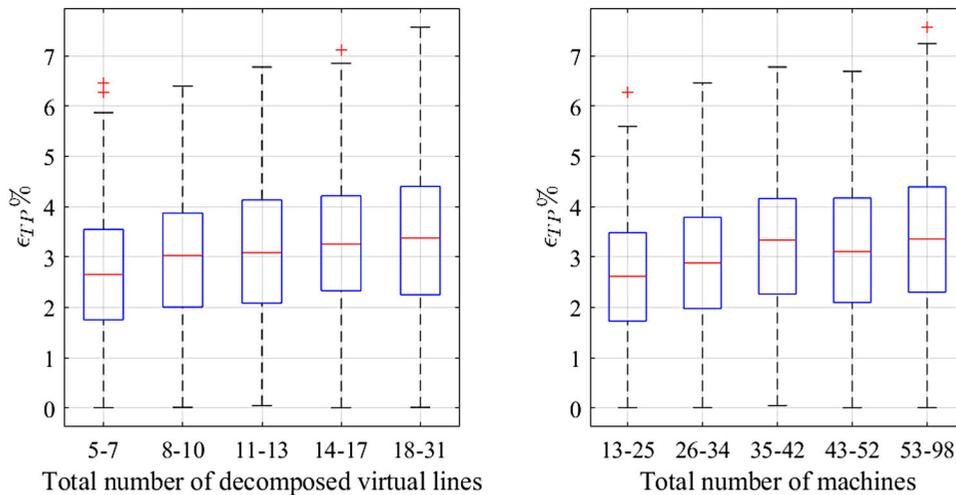


Figure 6. Dependence of ϵ_{TP} on system complexity.

level, denoted as α_i , is randomly selected from $\{2, 3, 4, 5\}$, and each component line is either an assembly component line or a disassembly one with equal probability $p = 0.5$. The total number of component lines in the system is

$$\gamma = \sum_{i=1}^r \alpha_i. \quad (23)$$

The number of machines in each of these γ component lines is randomly selected from $M_k \in \{4, 5, 6, 7, 8\}$, $k = 1, \dots, \gamma$. Next, for each component line level $k = 1, \dots, r - 1$, we randomly select $\beta_k \in \{2, 3, \dots, \alpha_{k+1}\}$ machines to be merge/split operations and randomly assign the α_{k+1} component lines in the $(k + 1)$ -th level to these merge/split operations. Clearly, some component lines may contain one or multiple merge/split operations, while some may have none. Similarly, the number of merge/split operations in the main line is randomly selected from $\beta_0 \in \{2, 3, \dots, \alpha_1\}$, which are randomly assigned to the α_1 first-level component lines. Finally, the total number of machines in the main line is randomly selected from $M_0 \in \{3\beta_0, 3\beta_0 + 1, \dots, 4\beta_0\}$, and the locations of the β_0 merge operations are randomly selected from the M_0 machines. The parameters of machines are randomly selected from (19)-(20).

For all 6,000 assembly systems constructed, the parameters of the virtual merge operations converge and the algorithm stops at terminal condition $\|\widetilde{TP}(s) - \widetilde{TP}(s - 1)\| < \epsilon$, where $\epsilon = 10^{-5}$. Therefore, we claim that RDAA II is convergent.

3.3. Performance metric calculation and accuracy

Similar to the single-merge-operation multiple-component-line systems, one can calculate the performance metrics for each virtual line using the aggregation procedure of Bai et al. (2020) and use them to approximate those of the original assembly system. Specifically, system throughput can be approximated by the average throughput of all ADUs: $\widetilde{TP} = \text{mean}(\widetilde{TP})$, while the work-in-process and machine starvation and blockage probabilities can be approximated by those of the corresponding buffer and machine in the virtual lines.

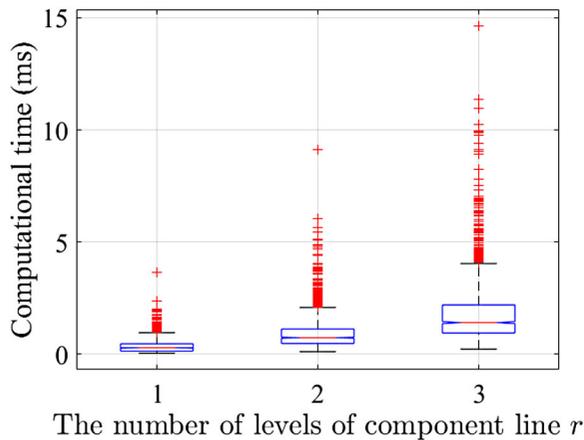
To study the accuracy of the performance metric approximation, the same 6,000 A/D systems generated in Subsection 3.2 are used again. Simulation with the same configuration as in Subsection 2.6.2 is used to estimate the values of the systems' performance metrics. The approximation errors and computational times of RDAA II, compared to simulation, are summarised

in Table 2, where we also list the average number of machines ($\sum M_i$), the average number of merge/split operations ($\sum MSO$), and the average number of virtual lines ($\sum VL$) for each r considered. One can see that RDAA II performs robustly and can maintain a good approximation accuracy even when the system size grows quite large (about 62 machines in total on average for $r = 3$). Moreover, the box plots of Figure 6 imply that the approximation error tends to increase (but at a diminishing rate) as the system complexity increases. We also notice that RDAA II tends to overestimate TP , compared to the simulation results. Among the 6,000 A/D systems studied, RDAA II overestimates TP in 98.2% of all cases by an average factor of 3.29%. For the remaining 1.8% of cases, where TP is underestimated, the average error is 0.63%. However, no such tendency for WIP approximation of RDAA II is observed. Results show that around half of the WIP s are overestimated, while the other half are underestimated. Specifically, RDAA II overestimates 51.25% of the WIP s by an average factor of 2.86%, and the remaining 48.75% buffers have underestimated WIP s by RDAA II with the average error at 2.67%. Detailed numerical case studies of the accuracy of RDAA II performance metrics approximation, in comparison with simulation, are provided in Appendices 1 and 2.

Next, we study the computational efficiency of RDAA II. Table 2 and Figure 7 show that although the computational time increases as the system complexity, the algorithm still converges quite fast: requiring only 1.8 ms on average for $r = 3$ and no longer than 14.65 ms for all cases studied. On the other hand, it takes nearly 2 minutes on average for the simulation to complete when $r = 3$. Note that similar results about the computation time are presented in Yegul et al. (2017) where it takes around 3 minutes for commercial simulation software, Simul8, to simulate a 24-workstation production system. While this computational time may still be acceptable for a single case calculation, it may quickly become impractical in optimization tasks, which typically require the evaluation of hundreds or thousands of alternative solutions, (see Yegul et al. 2017; Gansterer, Almeder, and Hartl 2014; Spieckermann et al. 2000). In that case, evaluating 10,000 system alternatives using simulation will cost about $(100 \text{ seconds/evaluation} \times 10,000 \text{ evaluations}) \div 86,400 \text{ seconds/day} = 11.5 \text{ days}$, while using the RDAA II algorithm only takes about $0.0018 \text{ seconds/evaluation} \times 10,000 \text{ evaluations} = 18 \text{ seconds}$. Therefore, developing computationally efficient algorithms for system performance evaluation is critical to analyzing and optimizing manufacturing processes in practical settings.

Table 2. Average approximation accuracy and computational efficiency of RDAA II.

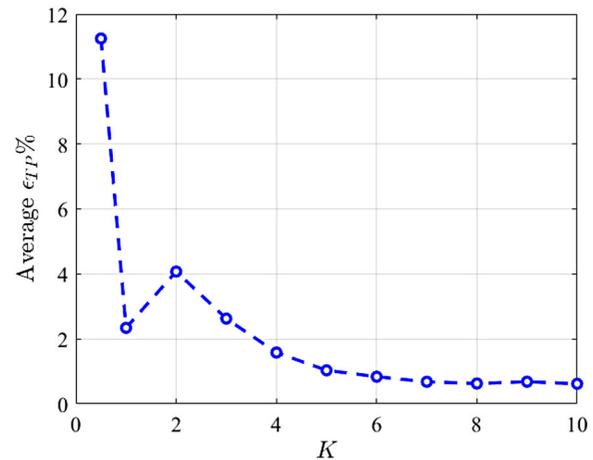
	Approximation error				Average system size			Average computing time (ms)	
	ϵ_{TP}	ϵ_{WIP}	ϵ_{ST}	ϵ_{BL}	$\sum M_i$	$\sum MSO$	$\sum VL$	RDAA II	Simulation
$r = 1$	3.03%	2.53%	0.0138	0.0138	30	3	7	0.36	62,842
$r = 2$	3.34%	2.63%	0.0149	0.0148	52	6	14	0.92	74,894
$r = 3$	3.33%	2.98%	0.0164	0.0164	73	8	20	1.8	100,266

**Figure 7.** Boxplot of computational efficiency of RDAA II.

Finally, we study the effects of buffer capacities on the approximation accuracy of RDAA II. To conduct the experiments, we randomly generate 100 different A/D systems. The system parameters except the buffer capacity are randomly selected from (19). For each system, thus generated, the buffer capacity N_{ki} is decided by the follow:

$$N_{ki} = \lceil K \cdot \min(T_{down,ki}, T_{down,ki+1}) \rceil, \quad (24)$$

where K is identical for all buffers in the system. Then, we vary K from $K \in \{0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. For each value of K , the average ϵ_{TP} of all 100 A/D systems is calculated and presented in Figure 8. The results show that RDAA II tends to have lower ϵ_{TP} as the buffer capacity increases (i.e. a larger value of K). This is due to less coupling among the machines, component lines, and AD units under larger buffers. A similar phenomenon is observed for the serial line aggregation algorithm developed in Bai et al. (2020) as well. Additionally, although RDAA II has a larger approximation error with small buffers (i.e. when $K = 0.5$), the accuracy improves to within 4% on average as soon as K reaches 1. Note that K indicates the number of average downtimes a buffer can accommodate and is typically selected as greater than 1 on the factory floor. Therefore, we claim that RDAA II is an effective algorithm for performance metrics calculation under practical system parameter ranges.

**Figure 8.** Effects of average buffer capacity on ϵ_{TP} .

3.4. Comparison with the decomposition technique

Both RDAA I and RDAA II are *recursive aggregation*-based algorithms for production systems performance evaluation. As explained in Subsection 3.1, the idea of RDAA II is to first decompose the entire system into several virtual serial production lines by introducing the virtual MSOs, then to use the aggregation method derived in Bai et al. (2020) along with Equations (2)–(17) to iteratively solve the parameters of the virtual MSOs by explicitly incorporating the effects from other parts of the system until convergence is observed, and finally to evaluate performance metrics of the original system from the counterparts in the virtual serial lines.

The other class of production systems performance evaluation algorithm (see Subsection 1.2), the *decomposition technique*, uses a different approach. Specifically, for A/D systems, as explained in Gershwin and Burman (2000), the decomposition approach directly decomposes the entire system into $|B|$ virtual two-machine lines, where $|B|$ is the total number of buffers. This results in six parameters (i.e. breakdown rates, repair rates, and processing speeds of the upstream and downstream machines) to be determined for each virtual two-machine line thus constructed. Then, based on the conservation of flow, flow rate-idle time, resumption of flow, interruption of flow, and boundary conditions, $6|B|$ equations are derived to capture the quantitative relationship

Table 3. Comparison of RDAA II with E-method from Gershwin and Burman (2000) and GE-method from Xia et al. (2016).

	$\epsilon_{TP}(\%)$	$\epsilon_{WIP}(\%)$	Standard deviation of ϵ_{WIP_i}
E-method	2.17	12.73	10.48
GE-method	1.68	13.37	10.58
RDAA II	0.71	5.47	4.50

among the virtual two-machine lines and the parameters involved. These equations are solved using iterative techniques. Finally, the performance metrics of the entire A/D system are evaluated based on the virtual two-machine lines.

Unfortunately, for A/D systems, neither our proposed algorithms, RDAA I/II, nor the decomposition technique-based algorithms in the literature offer any analytical results on algorithm convergence, computational complexity, etc. Therefore, as an alternative, numerical experiments are conducted in this subsection to compare the performance of RDAA II with two existing decomposition-based methods in the literature (one is the so-called *E-method* developed in Gershwin and Burman 2000 and the other is the *GE-method* proposed in Xia et al. 2016). Specifically, 50 cases are randomly generated from the same parameter ranges and system structure as used in paper (Xia et al. 2016). The results are summarised in Table 3. One can see that RDAA II significantly outperforms the other two methods in both *TP* and *WIP* approximation and have much better consistency in ϵ_{WIP_i} . The approximation accuracy of *ST* and *BL* is not compared since neither E-method nor GE-method provides formulas to calculate these quantities.

Therefore, we claim that RDAA II is an accurate and computationally efficient tool for performance metric calculation of general multi-level assembly/disassembly systems with exponential machines.

4. Industrial case study

In this section, we present an industrial case study to illustrate the application of the RDAA II algorithm developed above in a real manufacturing setting.

4.1. Background

The case study was conducted in a local company manufacturing brakes and clutches for customers from various application fields. The production facility was originally laid out in a job-shop-like manner, where machines were grouped together based on their functions (e.g. saw, lathe, milling). With a recent change in business operations, the management decided to evaluate the feasibility to create separate cells dedicated to its top-sale products. The

goal of this case study was to predict the throughput of the cell for one end product during the design stage and provide the information to the management for capacity planning, purchase/order scheduling, etc.

4.2. Data provided

To carry out the study, the company provided the academic team the bill of materials of the end product and its components (see Table 4) as well as the routings (i.e. the operation sequence, assigned machines, and the MTBF, MTTR, processing time of each operation) of all components made in-house (see Table 5). For confidentiality, the part information and the characteristics of the machines are masked and modified.

4.3. Modeling and system performance analysis

Based on the bill of materials and the production routing information given above, the manufacturing system layout can be illustrated by Figure 9. It consists of one end product (red shadowed part), six sub-assemblies (green shadowed part), and twelve input component lines (yellow shadowed part). Using the machine parameters given in Table 5 and RDAA II developed in this paper, we obtain its estimated throughput at $\widehat{TP} = 0.5285$ (jobs/min), while the throughput calculated by simulation is $TP^{sim} = 0.5193$ (jobs/min). The estimation error of four performance metrics and the number of iteration steps are provided below:

- $\epsilon_{TP} = 1.77\%$, $\epsilon_{WIP} = 1.94\%$, $\epsilon_{ST} = 0.0097$, $\epsilon_{BL} = 0.0116$;
- The number of iterations is 30.

Note that for this assembly system, it takes RDAA II about 2 ms to complete its calculation, while it takes 40 seconds for the simulation to finish. One can see that RDAA II works accurately and much more efficiently than simulation. The latter has been stressed by the plant operations manager as a critical feature as the time waiting for the simulation to finish every time a change is made to the system parameters can easily destroy his regular workflow.

Based on its product portfolio, the plant planned to allocate 10% of its production hours to this product, which amounts to 4 hours/week or 16 hours/month. This implies that the estimated throughput of the current system configuration is $\widehat{TP} = 0.5285(\text{jobs/min}) \times 960(\text{min/month}) \approx 507$ (jobs/month). According to the historical sales record, it was forecasted that the monthly demand will be about 550 (jobs/month). To close the gap

Table 4. Indented production bill of materials.

Level	Part	
0	End Product	
1	Sub assembly 2 (MO_2)	
2		Sub assembly 1 (MO_1)
3		Component 0 (main line)
3		Component 1 (CL_1)
3		Component 1 (CL_2)
2		Component 3 (CL_3)
2		Sub assembly 4 (MO_4)
3		Component 4 (CL_4)
3		Component 8 (CL_8)
1	Sub assembly 6 (MO_6)	
2		Sub assembly 5 (MO_5)
3		Component 5 (CL_5)
3		Component 9 (CL_9)
2		Sub assembly 7 (MO_7)
3		Component 10 (CL_{10})
3		Component 11 (CL_{11})
1	Component 6 (CL_6)	
1	Component 7 (CL_7)	

Table 5. Production routings and machine/buffer parameters.

	Op. seq.	Machine #	MTBF (min)	MTTR (min)	Cycle time (min)	Buffer
End Product	1	m_9	62.62	13.43	1.12	32
	2	m_{10}	50.86	10.84	1.19	–
Sub assembly 2	1	m_6	43.63	13.54	1.09	21
	2	m_7	94.91	12.60	0.84	17
	3	m_8	178.18	11.57	0.91	24
Sub assembly 1	1	m_3	49.57	12.03	0.93	15
	2	m_4	48.90	14.73	1.14	20
	3	m_5	47.92	13.66	0.90	38
Component Line 0	1	m_1	37.26	14.89	1.03	33
	2	m_2	93.80	12.97	0.88	13
Component Line 1	1	m_{11}	55.52	11.01	0.95	32
	2	m_{12}	40.15	12.74	1.18	30
Component Line 2	1	m_{13}	106.89	10.39	0.89	15
	2	m_{14}	146.48	13.64	1.13	14
Component Line 3	1	m_{15}	218.36	12.86	0.86	33
	2	m_{16}	151.64	14.81	1.09	24
Component Line 4	1	m_{17}	30.86	10.74	0.99	22
	2	m_{18}	32.33	12.88	0.87	32
Sub assembly 4	1	m_{19}	76.53	13.40	0.84	25
	2	m_{20}	169.79	14.03	0.94	25
Component Line 8	1	m_{29}	26.20	10.93	0.95	18
	2	m_{30}	200	10.67	1.14	28
Component Line 5	1	m_{21}	41.89	11.33	1.11	32
Sub assembly 5	1	m_{22}	75.57	14.99	0.97	24
Sub assembly 6	1	m_{23}	229.15	14.75	0.93	17
Component Line 6	1	m_{24}	69.05	13	0.84	33
	2	m_{25}	79.79	12.24	0.91	13
Component Line 7	1	m_{26}	50.49	11.10	0.90	15
	2	m_{27}	49.28	10.89	1.00	16
	3	m_{28}	52.90	12.98	1.12	24
Component Line 10	1	m_{33}	61.21	14.28	0.95	26
Sub assembly 7	1	m_{34}	102.94	10.80	0.99	29
Component Line 11	1	m_{35}	127.85	10.90	1.20	22
	2	m_{36}	45.90	14.47	0.96	30

between the current throughput capacity and the predicted monthly demand, the operations manager would like to have a list of operations/machines that should be improved and the amount of improvement needed to meet this target. To accomplish this, define machine i as the system's bottleneck (BN) machine if increasing its processing speed leads to the largest TP improvement,

i.e.

$$\frac{\partial TP}{\partial c_i} \geq \frac{\partial TP}{\partial c_j}, \quad \forall j \neq i. \quad (25)$$

Numerically, this can be implemented by evaluating the improvement in system throughput, ΔTP , when increasing the machine processing speed by the same amount

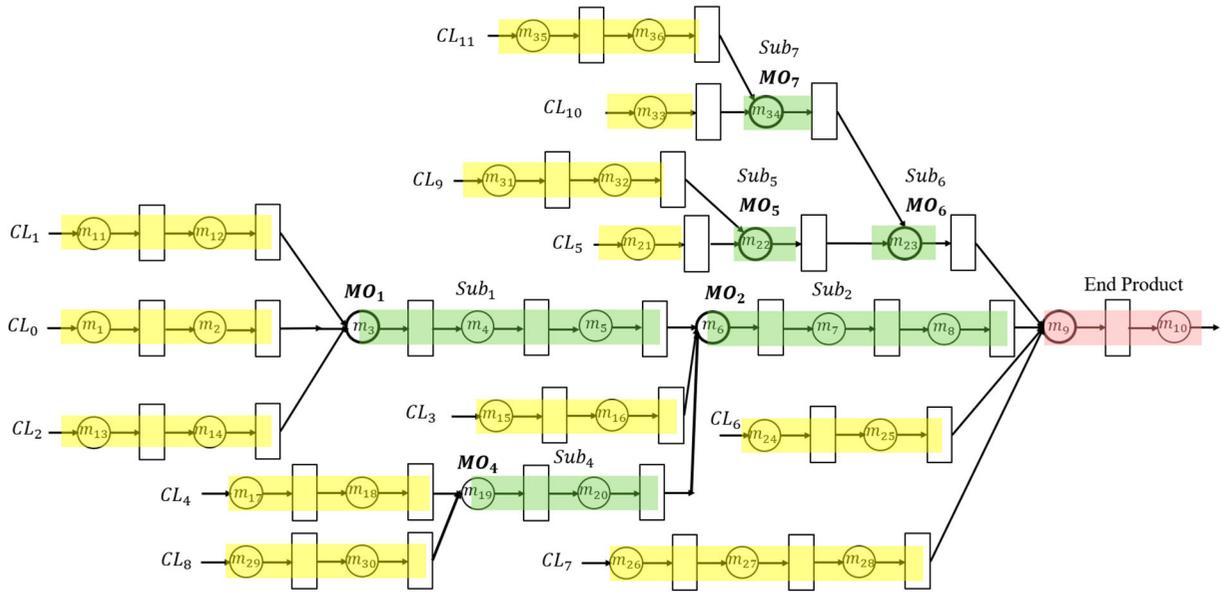


Figure 9. Production flow diagram of the industrial assembly system studied.

Δc one by one (and reverting to the original value before increasing the next machine’s parameter). Using RDAA II and with $\Delta c = 0.1$, we find that machine m_4 is the BN machine. Then, we improve c_4 to 0.9782 (jobs/min)

and obtain an improvement throughput $\widehat{TP} = 0.5448$ (jobs/min) or 523 (jobs/month). To ensure the throughput can eventually reach the target, we run a simulation on the modified system and obtain $TP^{sim} = 502$

Table 6. Improvement analysis of the industrial case to meet throughput target.

(a) BN machine #, improved TP (jobs/month) of each round and overall computational time					
Round	BN machine #	BN search by RDAA II		BN search by Simulation	
		Improved TP		BN machine #	Improved TP
		RDAA II	Simulation		
1	4	523	502	29	501
2	12	530	508	1	504
3	4	533	513	12	508
4	4	536	516	18	510
5	28	539	520	4	516
6	12	542	523	21	519
7	6	545	530	28	524
8	1	548	534	14	526
9	10	552	537	12	530
10	12	557	540	10	541
11	6	560	546	34	547
12	5	563	547	35	549
13	28	565	549	28	550
14	6	570	550		
Computational Time		9 min		260 min	
(b) Overall improvement					
Solution approach	RDAA II		Simulation		
System modification	<ul style="list-style-type: none"> • Increase c_1 from 0.97 (jobs/min) to 1.07 (jobs/min); • Increase c_4 from 0.88 (jobs/min) to 1.08 (jobs/min); • Increase c_5 from 1.12 (jobs/min) to 1.22 (jobs/min); • Increase c_6 from 0.92 (jobs/min) to 1.22 (jobs/min); • Increase c_{10} from 0.84 (jobs/min) to 0.94 (jobs/min); • Increase c_{12} from 0.84 (jobs/min) to 1.14 (jobs/min); • Increase c_{28} from 0.89 (jobs/min) to 1.09 (jobs/min); 		<ul style="list-style-type: none"> • Increase c_1 from 0.97 (jobs/min) to 1.07 (jobs/min); • Increase c_4 from 0.88 (jobs/min) to 0.98 (jobs/min); • Increase c_{10} from 0.84 (jobs/min) to 0.94 (jobs/min); • Increase c_{12} from 0.84 (jobs/min) to 1.04 (jobs/min); • Increase c_{14} from 0.89 (jobs/min) to 0.99 (jobs/min); • Increase c_{18} from 1.15 (jobs/min) to 1.25 (jobs/min); • Increase c_{21} from 0.90 (jobs/min) to 1.00 (jobs/min); • Increase c_{28} from 0.89 (jobs/min) to 1.09 (jobs/min); • Increase c_{29} from 1.05 (jobs/min) to 1.15 (jobs/min); • Increase c_{34} from 1.01 (jobs/min) to 1.11 (jobs/min); • Increase c_{35} from 0.84 (jobs/min) to 0.94 (jobs/min); 		

(jobs/month). Next, we repeat this process by identifying the BN of the modified system using RDAA II and improving the corresponding machine's processing speed until the monthly production target evaluated by simulation is reached. As a comparison, we also conducted this BN-based improvement design process using just simulation as the tool for both BN search and system throughput evaluation. The BN identified from each round of the search, the throughput improvement, total computational time, and overall improvement actions obtained by both RDAA II and simulation are summarised in Table 6.

These recommendations have been submitted to the plant to facilitate their cell design and workforce allocation on the floor. It should be noted that the solution process above, which involves an iterative search for system BNs, takes about 9 min using RDAA II and over 4.3 hours using just simulation.

As one can see from this industrial case study, the theoretical method developed above can indeed be used as an accurate and efficient tool to assist production practitioners in performing throughput prediction, capacity planning, system design, and other tasks.

5. Conclusion

In this paper, we propose two analytical algorithms (RDAA I and RDAA II) to calculate the performance metrics of assembly/disassembly systems with exponential reliability machines and finite capacity buffers. Specifically, RDAA I is designed for assembly systems with a single merge operation. It is based on the overlapping decomposition technique in Li (2005) and the aggregation procedure for serial exponential lines derived in Bai et al. (2020). Using the virtual serial lines constructed from overlapping decomposition and based on the different operating scenarios of the system, we derive the formulas to calculate the parameters of the virtual merge operations, which are the key to recreating (approximating) the parts flow of the original assembly system in the virtual serial lines. Then, the idea of RDAA I is extended to multi-level assembly/disassembly systems with general structures, leading to RDAA II, by decomposing a complex assembly/disassembly system into several assembly/disassembly units. The convergence, computational efficiency, and accuracy of both algorithms are justified using numerical experiments. An industrial case study is presented to demonstrate the practical applications of the algorithms developed. As demonstrated by the case study, the algorithms developed can serve as effective tools in system performance analysis, continuous improvement, design, and parameter optimization.

Future work in this area includes:

- Extension of the work to other production system structures (e.g. closed, parallel, and rework);
- Extension of the study to transient performance analysis, bottleneck identification, and system optimization;
- Implementing the algorithms into software packages for research, education, and practice;

Disclosure statement

No potential conflict of interest was reported by the author(s).

Notes on contributors



Yishu Bai received the B.E. degree in communication engineering from the North-eastern University at Qinhuangdao, Qinhuangdao, China, in 2017, and the M.S. degree from the Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL, USA, in 2018. She is currently pursuing the Ph.D. degree with the Smart Production Systems Lab, Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, USA. Her research interests include production systems modeling, analysis, improvement design, and visualisation of smart manufacturing systems.



Liang Zhang received the B.E. and M.E. degrees from the Center for Intelligent and Networked Systems, Department of Automation, Tsinghua University, Beijing, China, in 2002 and 2004, respectively, and the Ph.D. degree in Electrical Engineering-Systems from the University of Michigan, Ann Arbor, MI, USA, in 2009. He was with the Department of Industrial and Manufacturing Engineering, University of Wisconsin-Milwaukee from 2009 to 2013. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, USA. His research interests include modelling, analysis, improvement, design, control, and smart operations of manufacturing systems. He is a co-founder of Smart Production Systems LLC, Ann Arbor, MI, USA.

Data availability statement

The authors confirm that the data supporting the findings of this study are available within the article [and/or] its supplementary materials.

References

- Bai, Y., J. Tu, M. Yang, L. Zhang, and P. Denno. 2020. "A New Aggregation Algorithm for Performance Metric Calculation in Serial Production Lines with Exponential Machines: Design, Accuracy and Robustness." *International Journal of Production Research* 59: 4072–4089. doi:10.1080/00207543.2020.1757777.
- Bai, Y., and L. Zhang. 2021. "Performance Metrics Calculation for Assembly Systems with Exponential Reliability

- Machines.” 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi’an, China, 2021, 896–901. doi:10.1109/ICRA48506.2021.9561947.
- Chiang, S.-Y., C.-T. Kuo, J.-T. Lim, and S. Meerkov. 2000. “Improvability of Assembly Systems II: Improvability Indicators and Case Study.” *Mathematical Problems in Engineering* 6 (4): 359–393. doi:10.1155/S1024123X00001381.
- Chiang, S.-Y., C.-T. Kuo, J.-T. Lim, and S. Meerkov. 2000. “Improvability of Assembly Systems I: Problem Formulation and Performance Evaluation.” *Mathematical Problems in Engineering* 6: 321–357. doi:10.1155/S1024123X0000137X.
- Ching, S., S. M. Meerkov, and L. Zhang. 2008. “Assembly Systems with Non-exponential Machines: Throughput and Bottlenecks.” *Nonlinear Analysis: Theory, Methods & Applications* 69 (3): 911–917. doi:10.1016/j.na.2008.02.068.
- Colledani, M., A. Ratti, and C. Senanayake. 2015. “An Approximate Analytical Method to Evaluate the Performance of Multi-product Assembly Manufacturing Systems.” *Procedia CIRP* 33: 357–363. doi:10.1016/j.procir.2015.06.082.
- Dallery, Y., R. David, and X.-L. Xie. 1989. “Approximate Analysis of Transfer Lines with Unreliable Machines and Finite Buffers.” *IEEE Transactions on Automatic Control* 34 (9): 943–953. doi:10.1109/9.35807.
- Dallery, Y., and S. B. Gershwin. 1992. “Manufacturing Flow Line Systems: a Review of Models and Analytical Results.” *Queueing Systems: Theory and Applications* 12 (1–2): 3–94. doi:http://dx.doi.org/10.1007/BF01158636.
- Fowler, J. W., and O. Rose. 2004. “Grand Challenges in Modeling and Simulation of Complex Manufacturing Systems.” *Simulation* 80 (9): 469–476. doi:10.1177/0037549704044324.
- Gansterer, M., C. Almeder, and R. F. Hartl. 2014. “Simulation-based Optimization Methods for Setting Production Planning Parameters.” *International Journal of Production Economics* 151: 206–213. doi:10.1016/j.ijpe.2013.10.016.
- Gershwin, S. B. 1991. “Assembly/disassembly Systems: An Efficient Decomposition Algorithm for Tree-structured Networks.” *IIE Transactions* 23 (4): 302–314. doi:10.1080/07408179108963865.
- Gershwin, S. B., and M. H. Burman. 2000. “A Decomposition Method for Analyzing Inhomogeneous Assembly/disassembly Systems.” *Annals of Operations Research* 93 (1–4): 91–115. doi:10.1023/A:1018940310682.
- Helber, S. 1998. “Decomposition of Unreliable Assembly/disassembly Networks with Limited Buffer Capacity and Random Processing Times.” *European Journal of Operational Research* 109 (1): 24–42. doi:10.1016/S0377-2217(97)00166-5.
- Jacobs, D., and S. M. Meerkov. 1995. “A System-theoretic Property of Serial Production Lines: Improvability.” *International Journal of Systems Science* 26 (4): 755–785. doi:10.1080/00207729508929067.
- Jeong, K.-C., and Y.-D. Kim. 1997. “Performance Analysis of Assembly/disassembly Systems with Unreliable Machines and Random Processing Times.” *IIE Transactions* 30 (1): 41–53. doi:10.1023/A:1007441528431.
- Jia, Z., L. Zhang, J. Arinez, and G. Xiao. 2016. “Performance Analysis of Assembly Systems with Bernoulli Machines and Finite Buffers During Transients.” *IEEE Transactions on Automation Science and Engineering* 13 (2): 1018–1032. doi:10.1109/TASE.2015.2442521.
- Jia, Z., K. Zhao, Y. Zhang, Y. Dai, and C. Liu. 2019. “Real-time Performance Evaluation and Improvement of Assembly Systems with Bernoulli Machines and Finite Production Runs.” *International Journal of Production Research* 57 (18): 5749–5766. doi:10.1080/00207543.2018.1544426.
- Kumar, B. S., V. Mahesh, and B. S. Kumar. 2015. “Modeling and Analysis of Flexible Manufacturing System with Flexsim.” *International Journal of Computational Engineering Research* 5 (10): 1–6. doi:10.6084/M9.FIGSHARE.1605576.V1.
- Kuo, C.-T., J.-T. Lim, S. Meerkov, and E. Park. 1997. “Improvability Theory for Assembly Systems: Two Component-one Assembly Machine Case.” *Mathematical Problems in Engineering* 3. doi:10.1155/S1024123X97000501.
- Li, J. 2005. “Overlapping Decomposition: A System-theoretic Method for Modeling and Analysis of Complex Manufacturing Systems.” *IEEE Transactions on Automation Science and Engineering* 2 (1): 40–53. doi:10.1109/TASE.2004.835576.
- Li, J. 2013. “Continuous Improvement At Toyota Manufacturing Plant: Applications of Production Systems Engineering Methods.” *International Journal of Production Research* 51 (23–24): 7235–7249. doi:10.1080/00207543.2012.753166.
- Li, J., D. E. Blumenfeld, N. Huang, and J. M. Alden. 2009. “Throughput Analysis of Production Systems: Recent Advances and Future Topics.” *International Journal of Production Research* 47 (14): 3823–3851. doi:10.1080/00207540701829752.
- Li, J., and S. M. Meerkov. 2008. *Production Systems Engineering*. New York, NY: Springer Science & Business Media.
- Liu, X.-G., and J. A. Buzacott. 1990. “Approximate Models of Assembly Systems with Finite Inventory Banks.” *European Journal of Operational Research* 45 (2–3): 143–154. doi:10.1016/0377-2217(90)90181-A.
- Manitz, M. 2008. “Queueing-model Based Analysis of Assembly Lines with Finite Buffers and General Service Times.” *Computers & Operations Research* 35 (8): 2520–2536. doi:10.1016/j.cor.2006.12.016.
- Manitz, M. 2015. “Analysis of Assembly/disassembly Queueing Networks with Blocking After Service and General Service Times.” *Annals of Operations Research* 226 (1): 417–441. doi:10.1007/s10479-014-1639-x.
- Mascolo, M. D., R. David, and Y. Dallery. 1991. “Modeling and Analysis of Assembly Systems with Unreliable Machines and Finite Buffers.” *IIE Transactions* 23 (4): 315–330. doi:10.1080/07408179108963866.
- Mourtzis, D., M. Doukas, and D. Bernidaki. 2014. “Simulation in Manufacturing: Review and Challenges.” *Procedia Cirp* 25: 213–229. doi:10.1016/j.procir.2014.10.032.
- Neeraj, R., R. P. Nithin, P. Niranjhan, A. Sumesh, and M. Thenarasu. 2018. “Modelling and Simulation of Discrete Manufacturing Industry.” *Materials Today: Proceedings* 5 (11): 24971–24983. doi:10.1016/j.matpr.2018.10.298.
- Negahban, A., and J. S. Smith. 2014. “Simulation for Manufacturing System Design and Operation: Literature Review and Analysis.” *Journal of Manufacturing Systems* 33 (2): 241–261. doi:10.1016/j.jmsy.2013.12.007.
- Omogbai, O., and K. Salonitis. 2016. “Manufacturing System Lean Improvement Design Using Discrete Event Simulation.” *Procedia CIRP* 57: 195–200. doi:10.1016/j.procir.2016.11.034.

- Papadopoulos, H. T., and C. Heavy. 1996. "Queueing Theory in Manufacturing Systems Analysis and Design: A Classification of Models for Production and Transfer Lines." *European Journal of Operational Research* 92 (1): 1–27. doi:10.1016/0377-2217(95)00378-9.
- Shukla, O. J., G. Soni, and R. Kumar. 2021. "Simevents-based Discrete-event Simulation Modelling and Performance Analysis for Dynamic Job-shop Manufacturing System." *International Journal of Advanced Operations Management* 13 (2): 167–183. doi:10.1504/IJAOM.2021.116137.
- Spieckermann, S., K. Gutenschwager, H. Heinzl, and S. Voß. 2000. "Simulation-Based Optimization in the Automotive Industry-A Case Study on Body Shop Design." *SIMULATION: Transactions of The Society for Modeling and Simulation International* 75 (5/6): 276–286.
- Tekin, E., and I. Sabuncuoglu. 2004. "Simulation Optimization: A Comprehensive Review on Theory and Applications." *IIE Transactions* 36 (11): 1067–1081. doi:10.1080/07408170490500654.
- Wang, J.-Q., F.-Y. Yan, P.-H. Cui, T. Xia, F.-D. Cui, and S. Verwer. 2018. "Modeling and Analysis of Non-homogenous Fabrication/assembly Systems with Multiple Failure Modes." *The International Journal of Advanced Manufacturing Technology* 94 (9-12): 3309–3325. doi:10.1007/s00170-017-0785-0.
- Xia, B., B. Zhou, C. Chen, and L. Xi. 2016. "A Generalized-exponential Decomposition Method for the Analysis of Inhomogeneous Assembly/disassembly Systems with Unreliable Machines and Finite Buffers." *Journal of Intelligent Manufacturing* 27 (4): 765–779. doi:10.1007/s10845-014-0912-9.
- Yao, D. D. 1994. *Stochastic Modeling and Analysis of Manufacturing Systems*. New York, NY: Springer-Verlag.
- Yegul, M. F., F. S. Erenay, S. Striepe, and M. Yavuz. 2017. "Improving Configuration of Complex Production Lines Via Simulation-based Optimization." *Computers & Industrial Engineering* 109: 295–312. doi:10.1016/j.cie.2017.04.019.
- Zhang, L., C. Wang, J. Arinez, and S. Biller. 2013. "Transient Analysis of Bernoulli Serial Lines: Performance Evaluation and System-theoretic Properties." *IIE Transactions* 45 (5): 528–543. doi:10.1080/0740817X.2012.721946.

Appendices

Appendix 1. Assembly System Case Study

In this appendix, we present a detailed numerical study of the approximation accuracy of RDAA II for assembly systems. Specifically, a 19-machine assembly system structure, illustrated in Figure A1 is selected. The system consists of one main line and two sub-assembly component lines, where m_3 and m_5 are merge machines performing assembly operations. According to this layout and parameter ranges (19) and (20), we randomly generate 50 cases with different parameter configurations. As an illustration, the parameters of three example systems resulting from the above, denoted as S_1 , S_2 , and S_3 , are given in Table A1. Note that there is no buffer downstream of machine m_7 for the system structure considered here.

For the assembly systems generated above, we calculate their TP s and WIP s using RDAA II and simulation. For the three example systems, S_1 , S_2 , and S_3 , the detailed results of each individual WIP_i and the system TP s are summarised in Table A2. One can see that all three TP s are overestimated with approximation errors 3.56%, 1.86%, and 1.59%, respectively. In addition, among all 54 WIP_i 's (18 buffers for each case), 21 of them are underestimated with an average error of 1.55%. The remaining 33 WIP_i values are overestimated by an average factor of 3.22%.

For the overall 50 assembly systems studied in this appendix, Figure A2 shows the average and individual approximation errors of TP . As one can see, all 50 TP s are overestimated in this study and the average approximation error is 2.86%. Specifically, no ϵ_{TP} is above 5% except one (5.54%). Figure A3 presents the boxplots of the (absolute) ϵ_{WIP_i} , $i = 1, 2, \dots, 18$, of the 50 assembly systems. It can be seen that among all the 900 WIP values evaluated, most of the ϵ_{WIP_i} 's are no larger than 5% with only 24 extreme outliers (about 2.7% of the cases) having errors greater than 10%. Additionally, 433 (48.11%) of the WIP s are overestimated by an average factor of 2.48%. The remaining 467 (51.89%) WIP s are underestimated with an average approximation error of 2.60%.

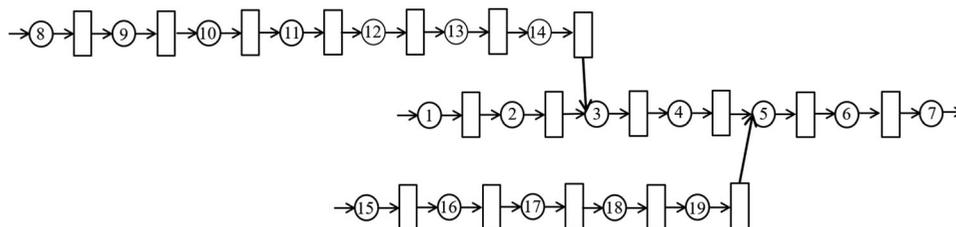


Figure A1. Assembly system considered.

Table A1. Example system parameters for the numerical study of RDAA II accuracy in assembly systems.

Machine	MTTR (min)			MTBF (min)			Cycle time (min)			Buffer capacity		
	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3
m_1	12.64	12.74	11.98	29.67	112.98	142.98	0.95	1.12	1.15	28	19	31
m_2	10.83	11.48	11.84	91.03	71.18	42.84	1.06	1.03	0.84	19	21	23
m_3	13.00	13.72	14.94	122.94	53.12	104.29	0.99	1.08	0.92	17	22	19
m_4	11.31	10.94	10.19	125.34	101.77	30.48	1.04	0.86	1.00	20	13	27
m_5	13.27	13.43	14.43	34.31	67.11	34.91	1.20	1.03	1.01	16	17	17
m_6	13.45	10.92	14.57	53.75	40.51	113.21	1.11	1.14	1.21	17	28	18
m_7	13.74	11.84	13.98	44.73	169.79	65.91	1.18	0.86	0.93	–	–	–
m_8	12.25	13.13	10.49	110.27	148.95	47.80	1.14	0.84	1.22	30	38	19
m_9	10.42	13.90	11.31	43.82	71.64	141.93	1.12	1.03	1.21	22	26	30
m_{10}	11.14	10.41	11.68	143.00	61.66	67.49	1.03	1.18	0.99	12	21	30
m_{11}	14.57	14.65	13.40	42.67	80.96	78.62	1.22	1.11	1.19	16	30	12
m_{12}	10.76	13.88	10.68	35.23	42.06	114.85	0.86	1.04	0.89	18	18	19
m_{13}	14.13	12.43	13.61	39.47	42.90	124.39	0.85	0.96	0.89	34	23	22
m_{14}	12.69	12.18	10.53	35.02	54.63	57.08	1.00	1.11	0.92	13	36	19
m_{15}	14.98	12.23	13.27	166.16	38.24	38.91	1.00	0.96	1.16	11	24	29
m_{16}	10.39	11.53	12.47	56.60	118.18	39.49	1.07	0.92	0.94	14	24	28
m_{17}	12.21	12.54	13.58	62.95	37.37	163.34	0.86	1.13	0.99	24	18	22
m_{18}	10.53	12.55	13.58	29.40	39.00	32.79	1.06	1.18	0.84	26	25	25
m_{19}	14.81	14.09	14.52	62.95	37.37	163.34	1.18	1.09	0.94	25	32	43

Table A2. *TP* and *WIP* evaluation of the example assembly systems.

	S_1			S_2			S_3		
	Simulation	RDAA II	Error	Simulation	RDAA II	Error	Simulation	RDAA II	Error
WIP_1	21.12	20.69	−1.53%	15.98	15.79	−0.96%	25.19	24.18	−3.26%
WIP_2	15.80	15.44	−1.85%	17.59	17.44	−0.72%	18.47	18.21	−1.13%
WIP_3	10.20	11.39	7.03%	12.24	13.21	4.43%	14.58	14.96	2.04%
WIP_4	15.38	16.51	5.65%	9.58	10.68	8.50%	16.88	16.96	0.27%
WIP_5	5.77	6.16	2.40%	6.14	6.69	3.23%	6.76	7.32	3.30%
WIP_6	6.01	6.46	2.61%	0.97	0.95	−0.08%	3.07	3.07	0.00%
WIP_7	27.21	27.28	0.24%	36.74	36.59	−0.39%	11.19	11.93	3.91%
WIP_8	12.28	17.14	0.61%	22.17	22.25	0.31%	21.55	22.59	3.48%
WIP_9	10.72	10.76	0.36%	15.03	14.75	−1.33%	24.62	25.76	3.80%
WIP_{10}	6.89	6.50	−2.41%	22.76	22.32	−1.46%	5.46	5.64	1.57%
WIP_{11}	9.16	9.10	−0.31%	10.69	10.93	1.33%	12.56	13.61	5.52%
WIP_{12}	20.57	21.52	2.80%	14.30	14.99	3.02%	17.13	18.74	7.33%
WIP_{13}	6.49	6.98	3.72%	23.69	24.78	3.03%	14.47	15.03	2.92%
WIP_{14}	10.00	10.03	0.24%	18.70	18.45	−1.04%	14.06	14.14	0.27%
WIP_{15}	10.40	9.98	−2.97%	21.94	21.83	−0.45%	14.69	15.83	4.07%
WIP_{16}	20.42	20.36	−0.21%	11.18	11.15	−0.18%	14.54	15.69	5.20%
WIP_{17}	18.80	18.07	−2.83%	10.72	10.13	−2.34%	14.32	16.12	7.22%
WIP_{18}	21.52	21.06	−1.82%	12.09	10.40	−5.30%	30.39	32.88	5.79%
<i>TP</i>	0.4872	0.5044	3.56%	0.5170	0.5266	1.86%	0.5525	0.5618	1.59%

Appendix 2. A/D System Case Study

In this appendix, a detailed numerical study of the approximation accuracy of RDAA II for A/D systems is provided. As illustrated in Figure A4, a 23-machine A/D system structure is selected. The system consists of one main line, one 1st-tier assembly component line, one 1st-tier disassembly component line, one 2nd-tier assembly component line, and two 2nd-tier disassembly component lines. In this system structure, m_2 performs assembly operations, m_5 and m_{10} perform disassembly operations, while m_{12} performs both assembly and disassembly operations. Following similar procedures as Appendix 1, according to this system structure and parameter ranges (19) and (20), 50 cases with different parameter configurations are randomly generated. Specifically, three of them are selected as example systems, denoted as S_4 , S_5 , and S_6 . Parameters of these example systems are given in Table A3. Note that there is no buffers downstream of machines m_6 , m_{13} , m_{16} , and m_{19} in this system structure.

Next, we calculate the *TP*s and *WIP*s of the 50 A/D systems using RDAA II and simulation. The detailed results of each individual WIP_i and the system *TP*s for the three example systems are summarised in Table A4. It can be seen that all three *TP* values are overestimated by RDAA II with an approximation error of 2.39%, 0.23%, and 2.77%, respectively. Moreover, exact half of the *WIP*s are overestimated by an average factor of 2.16%, and the remaining half is underestimated where the average approximation error is 1.97%. For the overall 50 A/D systems studied in this appendix, Figure A5 shows the individual and the average approximation errors of *TP*, which indicate that the *TP*s of 2 out of the 50 cases are underestimated by RDAA II, while the remaining 48 *TP* values are overestimated. The average *absolute* approximation error of *TP* is 2.37% with none greater than 5%. Figure A6 presents the boxplots of the (absolute) ϵ_{WIP_i} of the 50 A/D systems. It can be seen that among 1,100 *WIP*s evaluated in the 50 cases, most of the ϵ_{WIP_i} 's are under 5% with only 14 extreme outliers (about 1.3%

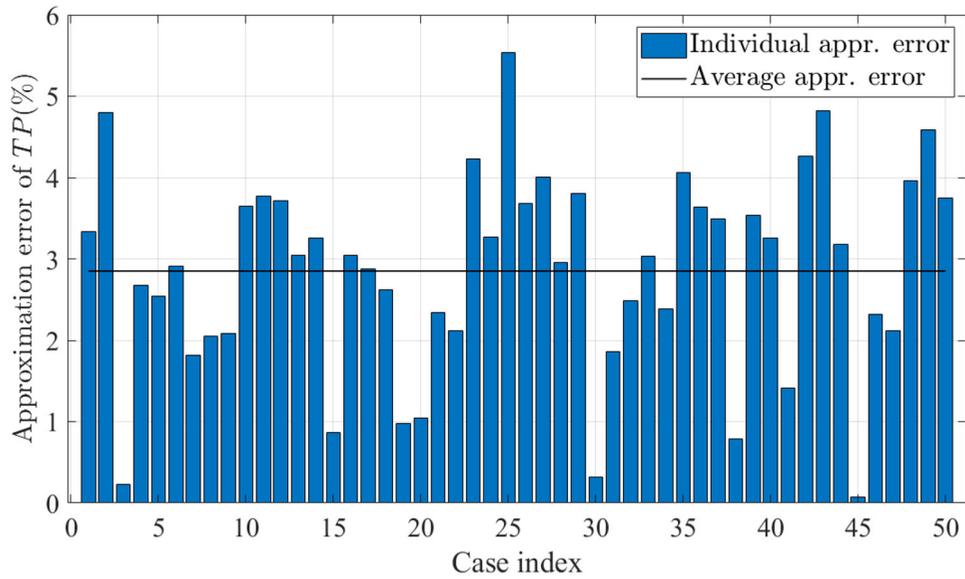


Figure A2. TP approximation error of assembly systems in this numerical study.

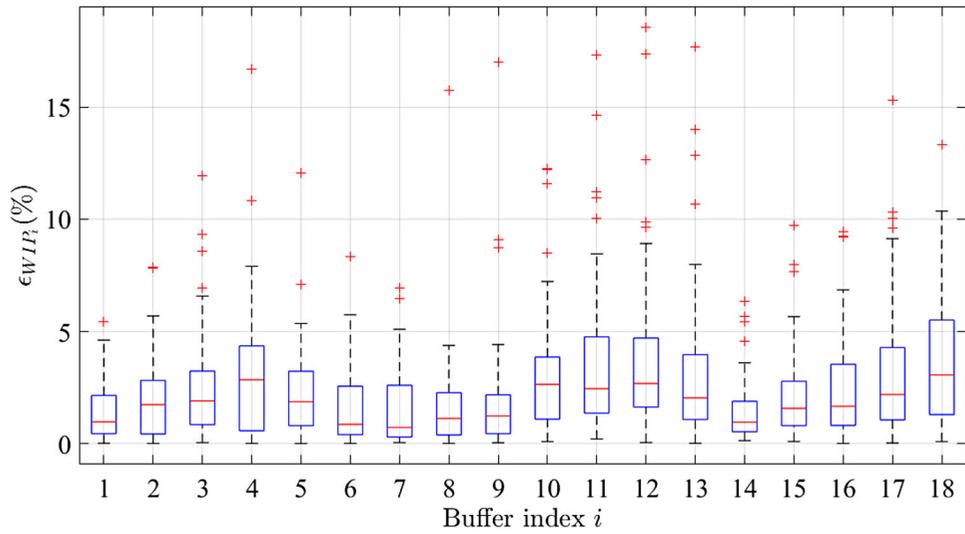


Figure A3. WIP approximation error of assembly systems in this numerical study.

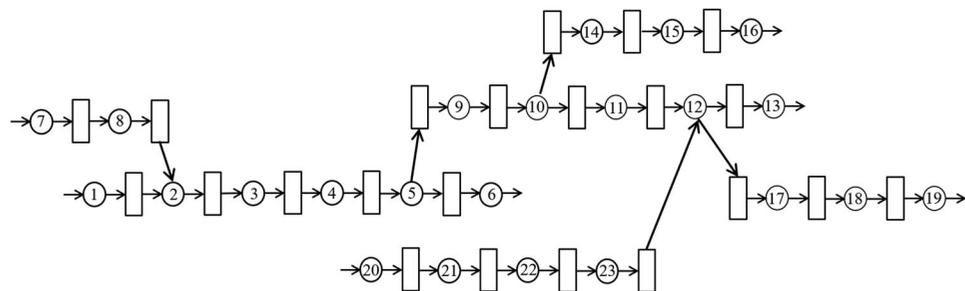


Figure A4. A/D system considered.

Table A3. Example system parameters for the numerical study of RDAA II accuracy in A/D systems.

Machine	MTTR (min)			MTBF (min)			Cycle time (min)			Buffer capacity		
	S ₄	S ₅	S ₆	S ₄	S ₅	S ₆	S ₄	S ₅	S ₆	S ₄	S ₅	S ₆
<i>m</i> ₁	12.73	11.30	10.30	128.11	51.85	62.87	0.90	0.93	0.90	22	19	29
<i>m</i> ₂	11.73	12.23	13.70	33.73	74.97	86.20	1.07	0.99	0.86	25	28	31
<i>m</i> ₃	13.11	14.22	12.53	62.06	167.88	136.96	1.12	1.21	0.90	31	16	17
<i>m</i> ₄	13.98	10.98	11.00	264.74	32.89	27.46	1.08	0.87	1.09	34	24	26
<i>m</i> ₅	13.73	11.52	12.14	51.29	45.75	114.56	0.97	1.07	1.16	27	25	14
<i>m</i> ₆	10.63	12.42	10.84	26.24	226.56	53.79	0.88	1.12	0.88	26	23	26
<i>m</i> ₇	14.11	11.69	13.76	43.12	46.93	95.08	1.09	1.18	0.90	–	–	–
<i>m</i> ₈	14.11	11.69	13.76	43.12	46.93	95.08	1.09	1.18	0.90	19	22	16
<i>m</i> ₉	10.13	13.99	11.84	40.37	89.44	36.12	1.04	1.08	1.10	29	13	15
<i>m</i> ₁₀	12.07	14.94	14.71	43.67	185.13	74.88	0.87	1.12	1.12	33	20	12
<i>m</i> ₁₁	13.66	10.80	10.09	42.64	200.24	71.01	0.96	0.94	1.08	21	13	27
<i>m</i> ₁₂	13.91	11.18	14.15	196.90	70.63	218.16	0.84	1.21	0.88	40	23	20
<i>m</i> ₁₃	11.84	13.51	13.13	79.77	36.00	56.40	1.13	1.10	0.89	27	21	32
<i>m</i> ₁₄	13.72	11.88	12.69	217.05	28.94	32.86	0.88	1.10	0.97	26	23	36
<i>m</i> ₁₅	14.46	14.87	13.25	61.45	87.34	33.14	0.93	0.87	0.97	23	37	27
<i>m</i> ₁₆	11.21	14.86	13.63	161.52	79.07	81.95	1.11	1.02	1.09	–	–	–
<i>m</i> ₁₇	10.65	13.22	10.47	25.02	208.91	93.26	1.01	0.91	0.93	13	39	31
<i>m</i> ₁₈	11.13	14.30	14.39	64.34	111.72	98.72	1.04	0.96	0.89	21	32	29
<i>m</i> ₁₉	11.75	12.01	10.07	106.07	77.37	37.06	0.96	0.90	1.02	–	–	–
<i>m</i> ₂₀	11.44	13.16	11.47	35.87	64.63	169.77	0.89	1.18	1.02	13	13	16
<i>m</i> ₂₁	14.64	14.93	10.90	204.21	48.58	53.23	1.19	0.84	1.01	32	17	30
<i>m</i> ₂₂	12.26	12.80	14.63	83.68	202.28	223.10	0.89	0.88	1.10	–	–	–
<i>m</i> ₂₃	12.96	14.67	10.34	125.85	74.26	26.38	0.88	1.22	0.93	28	25	12
<i>m</i> ₂₄	10.81	16.60	12.91	58.23	32.91	39.08	1.06	1.01	0.86	19	37	17
<i>m</i> ₂₅										28	35	30
<i>m</i> ₂₆										24	23	21

of cases) having errors greater than 10%. Additionally, 56.09% of WIPs are overestimated by an average factor of 2.45%, and

the remaining 43.91% of the WIP values are underestimated, in which the average approximation error is 2.22%.

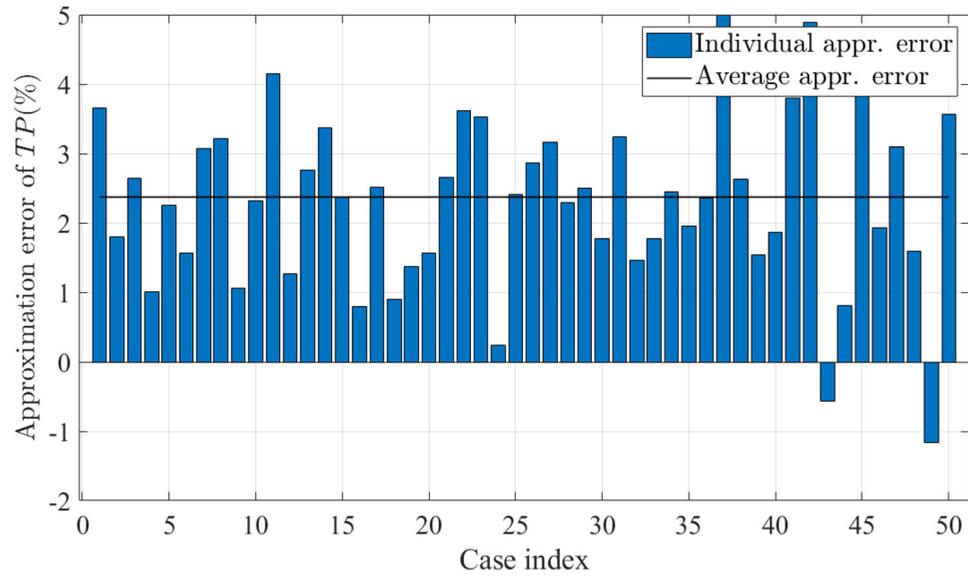


Figure A5. TP approximation error of A/D systems in this numerical study.

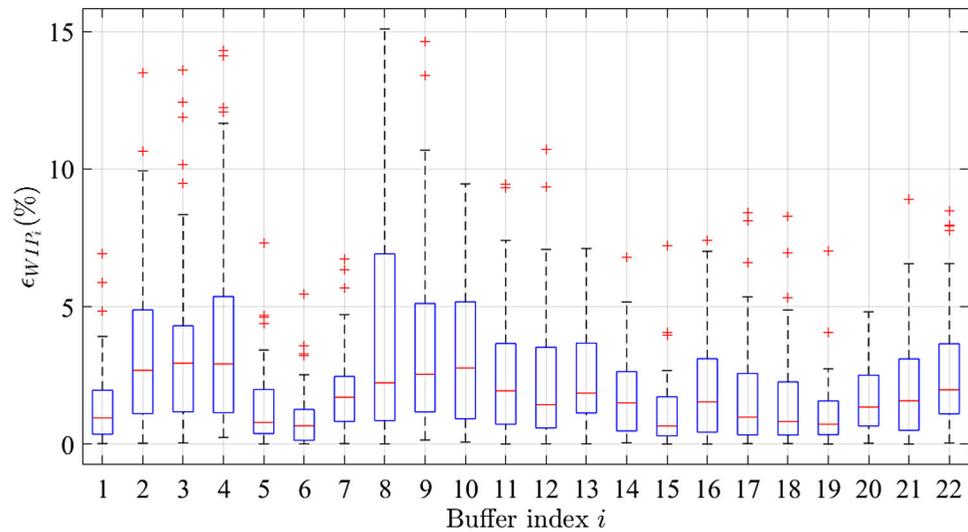


Figure A6. WIP approximation error of A/D systems in this numerical study.

Table A4. *TP* and *WIP* evaluation of the example A/D systems.

	S_4			S_5			S_6		
	Simulation	RDAAs II	Error	Simulation	RDAAs II	Error	Simulation	RDAAs II	Error
<i>WIP</i> ₁	20.37	20.32	-0.19%	15.93	16.13	1.06%	26.18	25.62	-1.90%
<i>WIP</i> ₂	9.13	8.51	-2.49%	21.06	22.02	3.42%	18.73	19.02	0.93%
<i>WIP</i> ₃	7.56	5.47	-6.73%	10.74	11.06	2.00%	13.80	14.18	2.24%
<i>WIP</i> ₄	16.18	14.09	-6.15%	17.56	18.60	4.31%	11.45	11.29	-0.64%
<i>WIP</i> _{5,6}	7.63	8.13	1.84%	1.57	1.58	0.03%	2.18	2.18	0.00%
<i>WIP</i> ₇	10.32	10.24	-0.42%	14.67	14.20	-2.13%	14.11	14.10	-0.10%
<i>WIP</i> ₈	16.63	15.99	-2.23%	8.83	9.32	3.77%	6.07	5.47	-3.99%
<i>WIP</i> _{5,9}	9.63	8.56	-4.14%	15.53	16.19	2.91%	13.69	12.73	-3.69%
<i>WIP</i> ₉	15.37	14.23	-3.47%	15.67	16.83	5.80%	5.50	5.37	-1.07%
<i>WIP</i> _{10,11}	3.61	2.89	-3.43%	11.35	11.90	4.22%	4.77	4.48	1.07%
<i>WIP</i> ₁₁	11.13	11.86	2.70%	15.10	15.15	0.21%	13.83	15.03	3.75%
<i>WIP</i> ₁₂	1.13	1.12	-0.03%	8.60	8.98	1.64%	10.65	12.64	5.53%
<i>WIP</i> _{10,14}	14.19	14.08	-0.28%	2.66	2.54	-0.49%	8.30	7.99	-1.55%
<i>WIP</i> ₁₄	6.10	6.37	2.07%	5.05	5.24	0.49%	6.41	7.05	2.07%
<i>WIP</i> ₁₅	10.01	9.61	-1.92%	1.04	1.17	0.41%	1.97	2.15	0.61%
<i>WIP</i> _{12,17}	5.41	5.13	-1.21%	5.00	4.57	-1.17%	5.83	7.02	4.40%
<i>WIP</i> ₁₇	2.12	1.87	-1.87%	3.38	2.80	-4.42%	6.10	7.09	6.17%
<i>WIP</i> ₁₈	6.59	6.58	-0.04%	6.35	6.29	-0.33%	1.72	1.71	-0.03%
<i>WIP</i> ₂₀	21.77	21.95	0.64%	20.52	20.79	1.11%	8.92	8.90	0.21%
<i>WIP</i> ₂₁	16.30	16.85	2.92%	35.14	35.40	0.70%	12.58	12.25	-1.93%
<i>WIP</i> ₂₂	25.52	25.79	0.97%	26.21	25.63	-1.64%	18.77	17.71	-3.52%
<i>WIP</i> ₂₃	18.77	19.03	1.14%	14.70	14.60	-0.47%	13.33	13.04	-1.42%
<i>TP</i>	0.5723	0.5860	2.39%	0.5153	0.5165	0.23%	0.5578	0.5733	2.77%