

Received 6 February 2024, accepted 1 April 2024, date of publication 5 April 2024, date of current version 15 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3385428

## RESEARCH ARTICLE

# Joint Allocation of Buffer Sizes and Service Times in Unreliable Assembly-Disassembly Systems

**KHELIL KASSOUL**  AND **NAOUFEL CHEIKHROUHOU** , (Senior Member, IEEE)

Geneva School of Business Administration, University of Applied Sciences Western Switzerland (HES-SO), 1227 Geneva, Switzerland

Corresponding author: Khelil Kassoul (khelil.kassoul@hesge.ch)

This work was supported by the Swiss National Science Foundation under Grant 100018\_182244.

**ABSTRACT** This paper introduces a modeling methodology for Assembly/Disassembly (AD) systems, proposing an effective simulation-based optimization approach. The objective is to simultaneously determine optimal allocations of service times and buffer sizes within unreliable AD systems, ultimately aiming to maximize the production rate. Our method integrates a global search approach, Genetic Algorithm (GA), with a local search method, Finite Perturbation Analysis (FPA). Through this integration, we extract production rate gradient information based on system parameters, which is then incorporated into a stochastic optimization algorithm to simultaneously identify the best allocation of service times and buffer sizes. The proposed approach is applied to various instances of AD systems, and experiments are conducted to assess its performance. The results demonstrate the superior performance of the integrated GA-FPA method compared to the standalone GA and FPA methods, both in terms of convergence behavior and solution quality. The identified allocation patterns and the potential to expand our approach to other complex manufacturing systems can provide valuable insights for managers seeking to establish more sustainable and efficient assembly and disassembly systems.


**INDEX TERMS** Assembly/disassembly systems, finite perturbation analysis, genetic algorithm, simulation, simultaneous allocation.

## I. INTRODUCTION

A manufacturing system is a collection of machines, buffers for storage, conveyors, parts for transportation, and other components that work together to produce goods [1]. Production is the process through which an item is transformed to raise its worth [2]. A wide number of industrial systems may be represented as high-volume Assembly/Disassembly (AD) systems of unreliable machines and finite buffers [3]. In these systems, the machines are responsible for assembling, disassembling, or manufacturing items by specified designs. Generally, these systems provide products of different natures that can be items from disassembling operations, sub-products from assembling processes, or final products due to the combination of assembly, disassembly, and manufacturing. In the context of the circular economy, AD systems are purposefully designed to incorporate the reuse of used

products or their components in the creation of new ones, including the production of refurbished products. These systems are versatile, supporting various economic scenarios such as disassembly operations (e.g., reusing sub-products, recycling) and assembly operations (e.g., assembling electronic cards, constructing automobiles). Within these intricately interconnected manufacturing systems, random events like difficulties in manual operations, quality issues, machine breakdowns, and variations in processing times across diverse system points can lead to temporary halts in machine operations. These disruptions significantly impact the operational efficiency of manufacturing systems. Even minor changes in design specifications may result in considerable performance losses. For instance, in a workshop employing a No-Wait rule, where any stoppage of machines immediately affects the Production Rate (PR), the system exhibits a fragile dynamic response to accommodate random interruptions [4].

The term buffer size refers to dedicated buffer locations between machines, physical floor restrictions, or the capacity

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina .

of transportation and conveying systems. It is then essential to consider various scenarios where products are heavy or bulky, even while it may not be expensive to execute considerable variations in buffer capacity in some situations. Therefore, when designing production systems, it is crucial to consider buffer size and appropriate service time allocation for each machine. Optimizing these factors can reduce investment costs and increase the PR. Hence, exploring methods to optimize buffer space and service time allocations is advantageous. Indeed, the design and optimization of manufacturing systems have been the goal of several studies [5], [6], [7], where the Buffer Allocation Problem (BAP) has been a significant subject of these studies over the past 30 years. While there have been extensive studies on the design of manufacturing systems, there has been a notable scarcity of research dedicated to addressing the Simultaneous Buffer and Service Time Allocation Problem (BSTAP).

This paper proposes a hybrid methodology for AD systems that combines Genetic Algorithm (GA) and Finite Perturbation Analysis (FPA) techniques. The developed algorithm uses FPA as a local search to estimate the gradients of PR and GA as a global search to generate the input solutions. The diversification and exploration abilities of GA provide quick access to the areas of the (near-) optimal solutions. FPA is utilized for exploitation and intensification in these areas to improve solutions further. Moreover, this approach can converge in a single simulation run, decreasing the convergence time [8]. The combination of FPA and GA is one of this article's contributions when allocating buffer sizes and service times simultaneously in the case of AD systems.

The following sections of this paper are organized as follows: In Section II, we provide a review of the literature about AD systems and related works. Section III introduces the mathematical formulation and describes the solution strategy. The development of the optimization approach is provided in Section IV. Section V presents the numerical experiments carried out and the discussions of the results. Finally, conclusions and suggestions for future research directions are provided in Section VI.

## II. LITERATURE REVIEW

In addition to the detailed examination of serial manufacturing lines, AD systems have also been subject to extensive study and analysis [9], [10]. The literature widely recognizes allocation problems, including the BAP and STAP, for their challenging combinatorial complexity [11]. This literature review is organized into three subsections. Subsection II-A is focused on buffer allocation problems, considering assembly systems with reliable and unreliable machines. Subsection II-B is related to the simultaneous allocation of buffer capacities and service times in assembly disassembly systems. Subsection II-C highlights our contribution.

### A. BUFFER ALLOCATION IN RESEARCH STUDIES

BAP has been a central theme of several studies on the optimization and design of manufacturing systems. In a study

conducted by Powell and Pyke [12], the authors explore the domain of basic asynchronous reliable assembly systems characterized by variable processing times. They contribute by formulating heuristic rules, which are unveiled through simulation analysis, to enhance system performance. The study concentrates on assessing the sensitivity of the initial buffer's placement concerning parameters linked to processing time distributions. In a parallel effort, Fuxman [13] introduces an analytical model that tackles the determination of the optimal allocation of buffer storage in the assembly line with reliable machines. The author presents the Buffer Elimination Algorithm (BELA) as an efficient tool for identifying optimal buffer configurations. To measure the system's throughput, Levner's graph is utilized in the analysis. Yamada et Matsui [14] explore a design approach for assembly systems, taking into account demand fluctuations, lead time, and cost considerations. Employing an iterative simulation method, the study determines optimal buffer sizes at each station to minimize buffer and overflow costs, utilizing a complex optimization approach. Hemachandra and Eedupuganti [15] suggest an approach to enumerate the state space as a solution method for handling finite storage in queuing models for assembly systems. The study considers objectives such as minimizing Work In Progress (WIP) and maximizing the PR, all while determining optimal buffer configurations. Shaaban et al. [16] concentrate on assessing the performance of reliable, unpaced merging assembly lines distinguished by asymmetric buffer storage sizes. Through the use of simulations, the authors investigate various configurations, which include variations in mean buffer storages, and imbalanced buffer allocations. Their findings indicate that, in specific setups of merging lines, it is feasible to achieve equivalent throughput performance between asymmetrical buffer patterns and merging lines with balanced buffers. Conversely, statistically significant superior performance is observed in terms of average buffer length in numerous configurations. The study underscores the significant potential for cost savings associated with maintaining an average buffer level throughout the lifespan of the production system. This suggests that intentionally allocating asymmetrical buffers between stations may represent a worthwhile and strategic approach.

Bulgak [17] investigates the optimal allocation of inter-stage buffers in split-and-merge assembly systems with unreliable machines. The study employs simulation models and genetic algorithms to identify optimal buffer configurations. Additionally, the research proposes the use of simulation based on artificial neural networks to enhance computational efficiency in the optimization process. Chegade et al. [18] present an approach that deals with the BAP in assembly systems, employing a multi-objective solution strategy. The study employs a multi-objective ant colony optimization algorithm, incorporating Lorenz dominance in the optimization process. Through this approach, the research considers the simultaneous objectives of minimizing the total buffer size and maximizing the PR. Bertazzi [19]

examines a two-line assembly production system with breakdowns and delays, developing a model to establish the optimal buffer capacity allocations. An exact algorithm is proposed and applied to a real-world problem, aiming to minimize storage costs and costs associated with delays and breakdowns during the process. Alfieri et al. [20] introduce a novel row-column generation algorithm designed for the allocation of buffer sizes in flow lines and assembly/disassembly lines. This algorithm leverages the theoretical properties of the time buffer approximation. In a comparative analysis, the researchers benchmarked their proposed algorithm against a standard Linear Programming (LP) solver and a state-of-the-art Mixed-Integer Linear Programming (MILP) solver. The results indicate that, in most instances, the introduced algorithm outperforms the LP solver. Moreover, it demonstrates superior robustness compared to the MILP solver, especially in terms of computation time. These findings suggest the effectiveness and efficiency of the presented row-column generation algorithm for buffer size allocation across various line configurations.

### B. SIMULTANEOUS ALLOCATION BSTAP

The challenges and issues within the BAP still exist, and the complexity of the analysis is further heightened by the extension of the study to the simultaneous allocation of BSTAP. Subsequently, Tempelmeier [21] introduces a decomposition technique for the appropriate allocation of buffers in a real-world automobile body assembly shop. This technique takes into account both variable and deterministic service times. The author adopts an approach where an initial target PR is chosen to reduce the total buffer size. Subsequently, the entire buffer value is fixed to maximize the previously determined target PR. To achieve this target PR, the service duration of each station is minimized. This method offers a systematic approach to buffer allocation in the context of a practical assembly setting. Smith [22] investigates the allocation patterns in the context of merge topologies and small split to solve simultaneously the workload and buffer sizes allocation for various small queueing networks with reliable machines using a quadratic programming technique. The primary objective of the study is to ascertain whether there are discernible patterns in service time and buffer allocation, investigating whether these patterns persist or are absent. The allocation schemes identified in the study align with certain aspects reported in [23].

Spieckermann et al. [24] employ a genetic algorithm within a simulation model to tackle a real-world buffer planning problem in a vehicle body assembly station. The optimization objective of this study incorporates considerations for service time variability, buffer sizes, and deviations from the upper bounds for service durations. The primary aim of the optimization process is to minimize the objective function, achieved through the adjustment of service times at each station and the overall buffer sizes in use. This approach aims to enhance the efficiency of the assembly station by

optimizing both service times and buffer sizes in the face of practical operational challenges. The authors conduct a case study involving a vehicle manufacturer to assess the suggested methodology. They conclude, aligning with the car's planning engineers, that the design process may be improved by using simulation-based optimization.

Cruz et al. [25] utilize a multi-objective genetic algorithm to address the simultaneous goals of minimizing the total number of buffers, maximizing the throughput rate, and minimizing the overall server rate within general service queueing networks. The authors employ the generalized expansion method (GEM) to assess the algorithm's performance, with their focus not necessarily centered on designing allocation patterns. Through the integration of the GEM with a multi-objective evolutionary algorithm (MOEA), the researchers successfully generated informative Pareto curves. These curves serve as effective visualizations, revealing the intricate tradeoffs between throughput, total buffer allocation, and overall service allocation. Later, Cruz et al. [26] present an approach aimed at maximizing the PR by concurrently minimizing both buffer spaces and service times. The objective is to optimize the performance of generic finite queueing networks. To achieve this, the authors employ a GA to identify solutions for the total buffer storage and processing times. Additionally, SA is utilized to rearrange the allocation of buffers along the system. This involves restructuring the entire buffer while preserving the best allocations previously determined for service time. The combined use of GA and SA offers a comprehensive optimization strategy for enhancing the efficiency of single-server queueing networks. The authors evaluate their methodology using the suggested automobile assembly system of [24]. Their results demonstrate that the evolutionary algorithm may increase the throughput in a variety of situations.

### C. PAPER CONTRIBUTION

Existing literature on AD systems optimization has primarily focused on systems with reliable machines, focusing on determining buffer sizes (BAP). Despite these efforts, there is still a need to model and analyze these systems in the context of unreliable machines by allocating service times simultaneously on the machines and buffer sizes on the available areas, which is the goal of this article. The objective is to maximize the system production rate within specified total constraints for buffers and service times. The advantage lies in the concurrent selection of buffers and service times, enabling a higher PR without additional costs. Optimal allocations for both buffers and service times can contribute to enhanced system production rates. Considering the correlation between machine efficiency and upstream/downstream buffer sizes for each machine, simultaneous allocation exploits this correlation for a more effective design. This consideration is crucial due to the substantial costs associated with investment and strategic decisions. The challenge of allocating service times and buffer sizes for AD systems is intensified by the

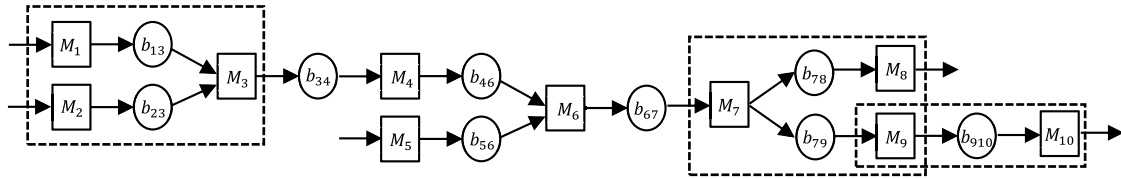


FIGURE 1. Example of an Assembly/Disassembly system.

vast search space, even for small or moderate-sized problems. Additionally, the absence of algebraic relations between production rates and buffer sizes further complicates the formulation of a mathematical model. To address these complexities, we extend the functionalities of GA-FPA, initially introduced by [27]. This expansion is designed to amplify the algorithm’s adaptability to contexts involving unreliable AD systems characterized by finite buffers and unreliable machines.

**III. PROBLEM DESCRIPTION AND SOLVING METHOD**  
**A. PROBLEM DESCRIPTION**

An AD system is conceptualized as an open queueing network, consisting of multiple machines (servers) distributed across various stages and interconnected by buffer spaces (refer to Fig. 1). In this configuration, each buffer serves as a connection point between precisely two machines (one downstream machine and one upstream machine). However, one machine connects two or more buffers [28]. An AD system can be deconstructed into three elementary cells: a transfer cell (Fig. 2a), a disassembly cell (Fig. 2b), and an assembly cell (Fig. 2c).

Fig. 1 shows an AD system comprising ten machines ( $M_1, \dots, M_{10}$ ) and nine buffers ( $b_{ij}$ ) ( $i=1, \dots, 9$  and  $j=2, \dots, 10$ ), where  $b_{ij}$  denotes the buffer sizes situated between the downstream machine  $M_j$  and the upstream machine  $M_i$ . The system is organized into six stages representing three connection points (corresponding to assembly or disassembly machines). An assembly (resp. disassembly) machine has two or more upstream (resp. downstream) buffers. In the given AD,  $M_3$  and  $M_6$  are assembly machines,  $M_7$  is a disassembly machine, and the remaining machines are transfer/production machines (i.e.,  $M_1, M_2, M_4, M_5, M_8, M_9$  and  $M_{10}$ ).  $M_1, M_2$  and  $M_5$  are the input machines from which the parts enter the system, and  $M_8$  and  $M_{10}$  are the output machines from which parts leave the system.

The following assumptions provide the general rules of functioning of the AD system:

- An AD system is a finite open queueing network with  $n$  unreliable machines ( $M_1, \dots, M_n$ ) alternating with  $m$  finite size buffers ( $\theta_1, \dots, \theta_m$ ), distributed on different stages and where  $\theta_i$  designates the buffer size situated between the downstream machine  $M_j$  and the upstream machine  $M_i$ ;
- Items evolve inside the system, which could produce different families of products, and each machine can treat several types of items;

- Each input machine seizes items from a buffer of infinite capacity in the network topology, and each output machine releases products into a buffer of infinite capacity.

In this study, we consider that the time of operational activity of a machine is composed of a deterministic part ( $\theta_{m+1}, \dots, \theta_{n+m}$ ) and a random part since the machines are subject to breakdown. We assume that the Mean Time Between Failures (MTBF) times and the Mean Time to Repair (MTTR) are geometrically distributed with respective probabilities of  $p_i$  and  $r_i$ . The process is assumed to it in a steady state (ergodic). Also, when a machine is operational (UP), it can be Full Output (FO) or blocked (resp. Null Input (NI) or starved) if at least one of its downstream (resp. upstream) buffers is full (resp. empty). These proprieties hold for transfer, assembly, and disassembly machines. We assume that a machine cannot fail when it is starved or blocked, and an input (resp. output) machine cannot be starved (resp. blocked). We define a potential FO “PFO” (resp. potential NI “PNI”) for a machine  $M_i$  as a state in which at least a downstream (resp. upstream) buffer holds  $b_{ij} - 1$  one (resp. one) unit, where  $b_{ij}$  denotes the buffer size situated between the downstream machine  $M_j$  and the upstream machine  $M_i$ . When a machine is blocked (resp. starved), no transit of parts is possible, all downstream (resp. upstream) stages are blocked (resp. starved), and no routing of parts is possible.

The PR of the system is calculated on all the output machines according to the following equation:

$$f(\theta) = \sum_{o \in O} f_o(\theta) \tag{1}$$

The design problem addressed in this paper aims to allocate  $B_{max}$  across  $m$  buffers and  $T_{max}$  across  $n$  machines to maximize the average PR  $f(\theta)$  of the AD system. The mathematical expression for this problem can be formulated as:

Find :  $\theta = (\theta_1, \theta_2, \dots, \theta_{n+m})$  to maximize  $f(\theta)$  (2)

Subject to :  $\sum_{i=1}^m \theta_i = B_{max}; \theta_i$  positive integer (3)

$\sum_{i=m+1}^{n+m} \theta_i = T_{max}; \theta_i \geq 0$  (4)

**B. SOLUTION METHOD**

The optimization approach proposed in this study is based on FPA [8], [27] coupled with GA [29], [30] and represented in Fig. 3. Indeed, combining GA with FPA aims to maximize the benefits of both techniques since the optimization and simulation are performed simultaneously, i.e., while GA (used as a global search) enables to reach quickly best solutions, FPA



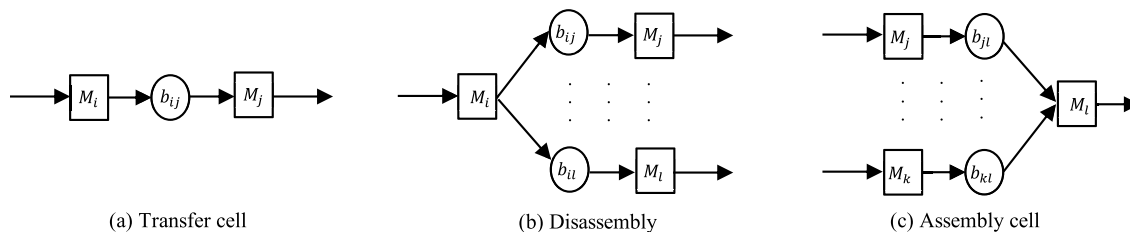


FIGURE 2. Elementary cells of an AD system.

may enhance the stochastic algorithm solutions by using a single run simulation to estimate the gradients  $\partial f(\theta)/\partial \theta_i$ .

Fig. 3 illustrates the comprehensive optimization technique, highlighting the main steps. The first step consists to generate the initial population,  $P = (r_1, r_2, \dots, r_p)$ , which contains  $p$  distinct solutions (configurations). Each configuration  $r_j = (\theta_{j,1}, \theta_{j,2}, \dots, \theta_{j,i}, \dots, \theta_{j,n+m})$  is obtained randomly and uniformly (We use the generation of the initial population as in [31]).  $\theta_{j,i}$  represents the  $i$ th variable decision of the  $j$ th configuration, where  $\theta_{j,i}$  ( $i = 1, 2, \dots, m$ ) denote the buffer sizes and  $\theta_{j,i}$  ( $i = m+1, \dots, n+m$ ) denote the service times. The PR of each individual  $r_j$  is assessed using a simulation model (Step 2). Following this, the operators of GA are used until the termination criterium is met (Step 3). A new population  $P' = (r'_1, r'_2, \dots, r'_p)$  is created from the existing population  $P$  by utilizing the operators of the GA across multiple generations to approach a nearly optimal solution (Step 4). The configurations of  $P'$  serve as the input for the stochastic algorithm procedure, which employs a gradient descent method [32] to estimate the gradients of PR  $\partial f(\theta)/\partial \theta_i$  relative to the decision variables (service times and buffer sizes)  $\theta_i$  ( $i = 1, \dots, n+m$ ) (Steps 5 and 6). The stochastic algorithm stops when the predetermined number of parts is achieved or when the PR does not improve with a new solution and a (near)optimal solution  $r^* = (\theta_1^*, \theta_2^*, \dots, \theta_{n+m}^*)$  is obtained (Step 7).

IV. METHODOLOGY

A. GENETIC ALGORITHM

In this study, the GA algorithm is utilized to generate initial solutions for the stochastic algorithm, aiming to obtain a nearly optimal solution. We apply tournament selection, an efficient and robust selection method commonly used by GAs [33]. Initially, two individuals are randomly selected from the current population, and their PRs, obtained by running a simulation model, are compared to determine the victor for the subsequent generation.

Subsequently, two parents (individuals),  $p_1$  and  $p_2$ , are selected from the current population and linearly combined using an arithmetic crossover operator [34] to generate two new offspring,  $o_1$  and  $o_2$ , using equations (5) and (6). In these equations,  $\alpha$  represents a coefficient chosen at random and uniformly from the range [1, 0]. It is important to highlight that an adjustment procedure is carried out to satisfy the

constraints (3), which corresponds to the mutation operator applied in this study.

$$o_1 = \alpha \cdot p_1 + (1 - \alpha) \cdot p_2 \tag{5}$$

$$o_2 = (1 - \alpha) \cdot p_1 + \alpha \cdot p_2 \tag{6}$$

Algorithm 1 outlines the procedure of the proposed genetic algorithm.

Algorithm 1 Genetic Algorithm

**Input:**  $p$  (number of configurations or individuals),  $n$  (number of machines) and  $m$  (number of buffers)

**Output:** Final population  $P'$

**Procedure:**

- a. **Generate** initial population  $P$  randomly and uniformly with  $p$  individuals.
- b. **For** each individual in  $P$ , **do**:
  - i. **Simulate** the individual using the Arena simulation model.
  - ii. **Calculate** the Production Rate (PR) based on the simulation results.
  - iii. **Assign** the calculated PR to the individual.
- c. **While** the size of the new population  $P'$  is less than  $p$  individuals, **repeat**  $P/2$  times:
  - **Select** randomly from  $P$  two parents and **use** the tournament selection to select the winner.
  - **Duplicate** the best (winner) individual.
  - **Cross** two parents,  $p_1$  and  $p_2$  to create two offspring by applying the arithmetic crossover operator.
  - **Apply** the adjustment procedure if the constraint (3) is not satisfied.
  - **Inject** the found offspring,  $o_1$  and  $o_2$ , in  $P'$ .

B. FINITE PERTURBATION ANALYSIS

Finite perturbation analysis is a method used to evaluate the effect of small, finite perturbations on a system’s behavior by studying its sample path [8]. This technique is commonly applied in control and dynamic systems to study the sensitivity of a system’s response to changes in its parameters or inputs, allowing the design of more robust and resilient systems. With FPA, the impact of perturbations on the performance of a system can be computed from a sample path or a Nominal Trajectory (NT) without additional experiments. The assumption here is that after a perturbation is introduced, subsequent interactions in the Perturbed Trajectory (PT) will exhibit statistical similarity to those in the NT. This means the PT does not need to be constructed or simulated, and the NT determines the system’s response to perturbations [8].

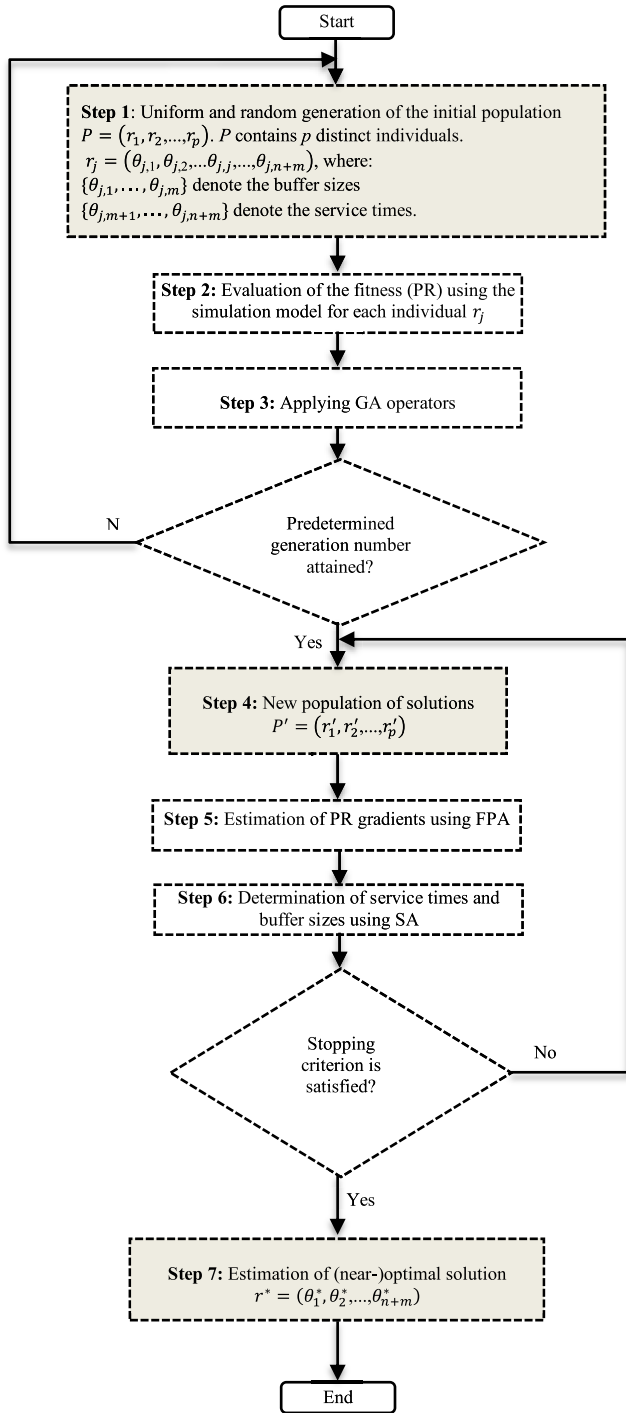


FIGURE 3. Global approach for the BSTAP.

In this study, the PR value is calculated by formulating perturbation generation and perturbation propagation rules which are intricately designed to anticipate the consequences of changes in buffer size in the NT, providing predictions, due to the introduction of perturbations, about the nature of future events and their respective durations. Earlier studies on serial manufacturing lines develop rules for generating and propagating perturbations in transfer/production cells [31],

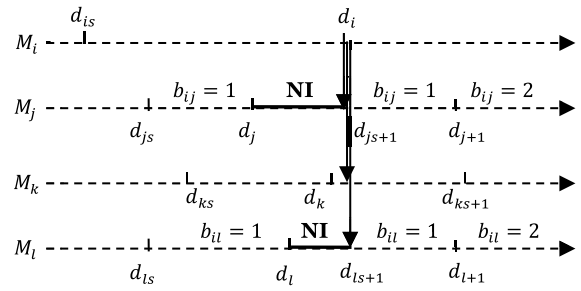


FIGURE 4. General disassembly process with NI periods.

[35]. In this study, we provide an overview of these rules as applied to transfer cells (Fig. 2a), and we also introduce new rules to extend their application to assembly/disassembly systems.

1) RULES FOR THE GENERATION AND PROPAGATION OF PERTURBATIONS IN TRANSFER CELLS

To analyze the system’s evolution in the PT, we suppose that Machine  $M_i$  (resp.  $M_j$ ) encounters a perturbation, denoted as  $\Delta t_i$  (resp.  $\Delta t_j$ ), which can be either propagated or generated. The overall value of the perturbation transmitted to Machine  $M_j$  is expressed as follows:

$$\Delta t'_j = (\text{new starvation finish time from the PT}) - (\text{old starvation finish time from the NT}) \quad (7)$$

Due to the introduction of this perturbation, several cases can be considered:

- Equation (8) summarizes all cases of PNI/ NI:

$$\Delta t'_j = \max\{\Delta t_i + [\text{NI}], \Delta t_j\} - \max\{0, [\text{NI}]\} \quad (8)$$

where [NI] represents a starvation interval time’s algebraic value for the machine  $M_j$ , denoted as  $\text{NI} = (d_i - d_j)$ .

- Equation (9) summarizes all cases of PFO/ FO periods:

$$\Delta t'_i = \max\{\Delta t_j + [\text{FO}], \Delta t_i\} - \max\{0, [\text{FO}]\} \quad (9)$$

where [FO] denotes the blocking interval time’s algebraic value (if it exists) for the machine  $M_i$ , denoted as  $\text{FO} = (d_i - d_j)$ .

2) RULES FOR THE GENERATION AND PROPAGATION OF PERTURBATIONS IN DISASSEMBLY CELLS

Consider a disassembly machine  $M_i$  (as represented in Fig. 2b) ending many NI periods at the same time on downstream stages. Fig. 4 shows the case when  $M_i$  terminates NI periods on  $M_l$  and  $M_j$  but no NI exists on  $M_k$  and no FO exists at any stage. Arrows indicate the direction of part releasing from one machine to the other. We define:

- $d_i$  as the nominal service end time of a particular service duration in machine  $i$ ;
- $d_j$  as the nominal service end time of a particular service duration in machine  $j$ ;

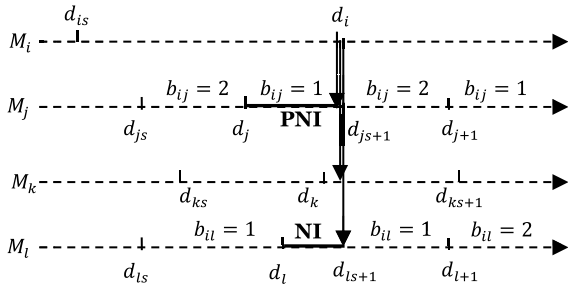


FIGURE 5. Sample nominal path of a disassembly cell with PNI.

- $d_{is}$  as the nominal service start time of a particular service duration in machine  $i$ ;
- $d_{js}$  as the nominal service start time of a particular service duration in machine  $j$ .

Three cases can be determined to predict the perturbed path of the overall tableau of events:

Case 1.  $d_i + \Delta t_i > d_j + \Delta t_j, \forall j$ .  $M_j$  is NI, the perturbation is fully propagated to every NI server, and the newly generated perturbation is given by:

$$\Delta t'_j = (d_i + \Delta t_i) - d_j = \Delta t_i \text{ and } \Delta t'_k = 0 \quad (10)$$

where  $M_k (k \neq j)$  is other than a blocked or starved server.

Case 2.  $d_i + \Delta t_i < d_j + \Delta t_j, \forall j$ .  $M_j$  is NI and terminated by specific service from  $M_i$ . In that case, all NI periods are eliminated, and we have partial propagation.

$$\Delta t'_j = (d_j + \Delta t_j) - d_i = \Delta t_j + (d_j - d_i) \text{ and } \Delta t'_k = 0. \quad (11)$$

for every NI server  $M_j$  and  $M_k (k \neq j)$  is other than a blocked or starved server.

Case 3.  $\exists M_{j_1}, \dots, M_{j_n}$  NI such that  $d_{j_l} + \Delta t_{j_l} < d_i + \Delta t_i < d_{j_k} + \Delta t_{j_k}, (k = 1, \dots, n \text{ and } k \neq l)$ . This case is a combination of the two first cases and results in:

$$\Delta t'_{j_k} = (d_{j_k} + \Delta t_{j_k}) - d_i = \Delta t_{j_k} + (d_{j_k} - d_i), k = 1, \dots, n. \quad (12)$$

$$\Delta t'_{j_l} = (d_i + \Delta t_i) - d_i = \Delta t_i, \quad k \neq l \quad (13)$$

- Case of PNI and PFO periods

A similar mechanism can be observed if some servers are PNI in the nominal path and may be converted in the perturbed trajectory into real NI. Consider the disassembly cell of Fig. 5, where the server  $M_j$  is PNI since the buffer  $b_{ij}$  holds a unique part inside.

Case 1.  $d_i + \Delta t_i > d_j + \Delta t_j$ . The PNI is converted into a NI period on  $M_j$ . A NI period on  $M_j$  is created because the perturbation is partially propagated:

$$\Delta t'_j = (d_i + \Delta t_i) - d_j = \Delta t_i + (d_i - d_j) \quad (14)$$

Case 2.  $d_i + \Delta t_i \leq d_j + \Delta t_j$ . All service event sequences maintain their perturbation, and the PNI remains PNI periods.

$$\Delta t'_j = (d_j + \Delta t_j) - d_j = \Delta t_j \quad (15)$$

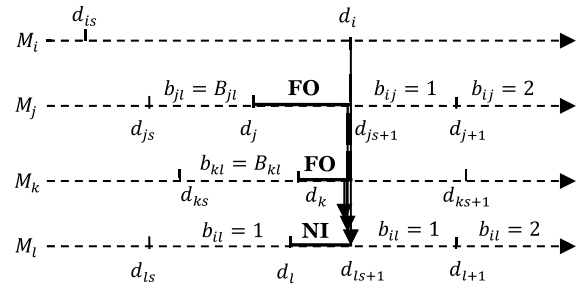


FIGURE 6. General assembly cell with FO and NI periods.

Similarly, the rules for perturbation propagation in the case of PFO periods are developed.

### 3) RULES FOR THE GENERATION AND PROPAGATION OF PERTURBATIONS IN ASSEMBLY CELLS

Analyzing assembly cells is interesting when FO periods exist at different stages. FO periods may eliminate gains on some servers, so the propagation stops. Consider an assembly cell (as represented in Fig. 2c), where  $M_l$  is an assembly machine, and the upstream machines  $M_k$  and  $M_j$  are FO due to the fullness of respectively  $b_{kl}$  and  $b_{jl}$ . We may focus our interest on the case where  $M_l$  is NI, when perturbations can be propagated.

Under the same assumptions, the end of the NI period on  $M_l$  is caused by the service end of at least one server say  $M_i$ , the other parts are retired from different buffers.  $M_l$ 's events sequence in a perturbed path is deduced from the sequence of the perturbed events on  $M_i$  (Fig. 6). If somehow perturbations are created or propagated on  $M_l$  and on different other stages, three cases have to be considered:

Case 1.  $d_l + \Delta t_l \leq d_i + \Delta t_i$ . The NI period on  $M_l$  remains and perturbation is fully propagated:

$$\Delta t'_l = (d_i + \Delta t_i) - d_l = \Delta t_i \quad (16)$$

Case 1.a) If there exists a FO period in the nominal path of  $M_j$  such that  $d_j + \Delta t_j > d_i + \Delta t_i$ , this period will be eliminated in the perturbed path and transformed into PFO. This process is referred to as backward propagation, in contrast to the previously mentioned forward propagation.

$$\Delta t'_j = (d_j + \Delta t_j) - d_i = \Delta t_j + (d_j - d_i) \quad (17)$$

Case 1.b) If there exists a FO period in the nominal path of  $M_j$  such that  $d_j + \Delta t_j < d_i + \Delta t_i$ , the FO period remains, and the perturbation is completely propagated:

$$\Delta t'_j = (d_i + \Delta t_i) - d_j = \Delta t_i \quad (18)$$

In other words, the determination of the new perturbation value is influenced by the machine states in parallel stages rather than solely relying on the perturbation value of  $M_l$ .

Case 2.  $d_l + \Delta t_l > d_i + \Delta t_i$ . The NI period on  $M_l$  is removed and the value of the new perturbation is:

$$\Delta t'_l = (d_l + \Delta t_l) - d_i = \Delta t_l + (d_l - d_i) \quad (19)$$

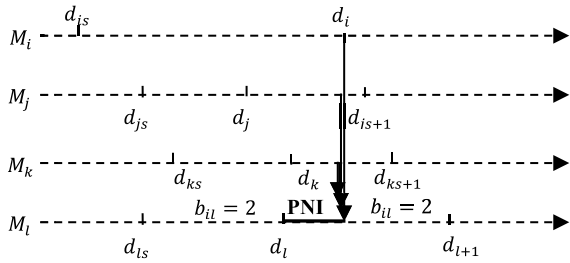


FIGURE 7. General assembly cell with PNI periods.

Note that in this case, the propagated perturbation value on  $M_l$  depends strongly on events on  $M_i$ . The case of the existence of FO periods on stages  $j$  is resolved through cases 1.a) et 1.b). We assumed that service on  $M_l$  depends only on a particular service end on  $M_i$  and even if  $d_l + \Delta t_l > d_i + \Delta t_i$ , the service beginning in the perturbed path will be at  $d_i + \Delta t_i$ , independent on  $M_l$  events.

• Case of PNI and PFO periods

To simplify the representation of the problem and its understanding, we consider an assembly cell where only one period PNI is present in the nominal path (Fig. 7). The stock  $b_{il}$  contains two parts until the instant  $d_i$ , when  $M_l$  charges a new piece. Thus,  $M_l$  enters a period of potential starvation, until  $d_i$ , when  $M_i$  serves the stock. Depending on the amplitude of perturbations carried by the different servers, two distinct cases can be identified:

Case 1.  $d_l + \Delta t_l < d_i + \Delta t_i$ . The perturbation on  $M_i$  is entirely transmitted to  $M_l$ . The PNI period on  $M_l$  remains in the perturbed trajectory.

$$\Delta t'_l = (d_i + \Delta t_i) - d_i = \Delta t_i \quad (20)$$

Case 2.  $d_l + \Delta t_l \geq d_i + \Delta t_i$ . The PNI period is eliminated and the perturbation on  $M_i$  is partially transmitted to  $M_l$ .

$$\Delta t'_l = (d_l + \Delta t_l) - d_i = \Delta t_l + (d_l - d_i) \quad (21)$$

Similarly, the rules for perturbation propagation in the case of PFO periods are developed. The system's evolution in a perturbed environment is predicted using these different rules. The prediction results from a single sample trajectory and relies on information about the state of each machine at every event. Collectively, these rules constitute the core of the developed FPA algorithm.

C. FPA ALGORITHM

1) BUFFER ALLOCATION ALGORITHM

Assuming that during a simulation (or an observation) of an AD system, each output machine  $M_o$  treats a total of  $L_o$  parts. We suppose that the simulation is long enough for the occurrence of all potential events. The PR (denoted by  $f$ ) calculated on the time  $T$  is given as:

$$f = \sum_{o \in O} L_o / T \quad (22)$$

where the index  $O$  represents all system output machines: Hence;

$$\partial f / \partial \theta_i = \partial \left( \frac{\sum_{o \in O} L_o}{T} \right) / \partial \theta_i \quad (23)$$

The simulation in this study operates under the assumption that the termination condition is determined by the overall parts number produced by all output machines in the system, i.e.,  $\sum_{o \in O} L_o = L$ .

$$\begin{aligned} \frac{\partial f}{\partial \theta_i} &= -L \cdot \left( \frac{1}{T^2} \right) \cdot \frac{\partial T}{\partial \theta_i} = -\left( \frac{L}{T} \right) \cdot \left( \frac{1}{T} \right) \cdot \frac{\partial T}{\partial \theta_i} \\ &= -\frac{f}{T} \cdot \frac{\partial T}{\partial \theta_i} \end{aligned} \quad (24)$$

where  $\partial T / \partial \theta_i$  is the duration of perturbation affecting the system after the generation of perturbation  $\Delta \theta_i$ . We note that in this work, a perturbation of one virtual unit buffer size is used for evaluating the PR gradient using the FPA propagation rules for each event. The persistence and the duration of an event along the disturbed trajectory are predicted concerning the repercussion of the perturbation. The value of  $\partial T / \partial \theta_i$  is determined using the FPA. Consequently, the change in total simulation time ( $\Delta T$ ) is computed for the output machine responsible for completing the  $L$  parts within the perturbed path. Hence, we can express this relationship as follows:

$$\begin{aligned} \frac{\partial f}{\partial \theta_i} &= -\frac{f}{T} \cdot \frac{\partial T}{\partial \theta_i} = -\frac{f}{T} \cdot \Delta T \\ &= -\frac{f}{T} \cdot (\text{Max}_{o \in O} \{d_{ofin} + \Delta t_o\} - T) \end{aligned} \quad (25)$$

Here,  $t_o$  represents the total duration of the perturbation accumulated after processing  $L_o$  parts by the output machine  $M_o$ . As such,  $\Delta t_o$  signifies the net gain or loss in time that is transmitted to machine  $M_o$  when a perturbation  $\Delta \theta_i$  is introduced, and it is computed using the FPA rules. The expression  $(\text{Max}_{o \in O} \{d_{ofin} + \Delta t_o\} - T)$  serves as an estimate for the actual value of  $\frac{\partial T}{\partial \theta_i}$ .

Algorithm 2 outlines a simplified FPA algorithm utilizing the established rules for buffer sizes. This algorithm computes the PR sensitivity vector concerning buffer sizes. We should note that incorporating simultaneous perturbations on various buffers, along with the concurrent sensitivity assessment, can be easily executed by introducing the sum of different buffers into the realized (global) gain.

2) SERVICE TIMES ALGORITHM

In this subsection, we present FPA for service times. With slight modification, we use the algorithm developed by [36] for cycle time optimization of closed-loop flexible assembly systems. We use the perturbation analysis gradient estimator to calculate  $\partial f / \partial \theta_i$  (for  $i = 1, \dots, n$ ). The main procedures of the algorithm are summarized in three steps.

Step 1 Initialization of accumulators  $A_{ij}$

The initial step in calculating the gradients involves initializing the accumulator variables, denoted as  $A_{ij}$ . These



**Algorithm 2** FPA for Buffer Sizes

**Perturbations generation**

**Step 1**

**If**  $M_i(t)$  is FO,  $i = 1, \dots, n$  **then**  
 $Sum_i = 0$   
 (gain of time on machine  $i$ )  
 Select  $b_{ij}$  in  $(I; J) = \{1, \dots, n - 1; 2, \dots, n\}$   
 (initialization of accumulators  $Sum_i$ )

**Step 2**

( $b_{ij}$  is the buffer located between machine  $i$  and machine  $j$ )

**If**  $M_i(t)$  is FO for the first time **then**  
 $b_{ij} = b_{ij} + \Delta b_{ij}$  (in this paper  $\Delta b_{ij} = 1$ )  
 $Sum_i = t_i \cdot \Delta b_{ij}$  ( $t_i$  is the processing time of the machine  $i$ )

**Perturbation propagation**

**Step 3**

**Case 1. If**  $M_i$  is a transfer machine **then**

**Step 3.a**

**If**  $M_j(t)$  is PNI/NI **then**  
 $Sum_j = \max \{Sum_i + [NI], Sum_j\} - \max \{0, [NI]\}$

**Step 3.b** (not applicable to the stock's first variation)

**If**  $M_i(t)$  is PFO/FO **then**  
 $Sum_i = \max \{Sum_j + [FO], Sum_i\} - \max \{0, [FO]\}$

**Case 2. If**  $M_i$  is a disassembly machine **then**

**If**  $M_j(t)$  is PNI (resp. NI, PFO, FO) **then**

propagate perturbation using elaborated corresponding rules (IV.B.2)

**Case 3. If**  $M_i$  is an assembly machine **then**

**If**  $M_j(t)$  is PFO  $\gg$  (resp. FO, PNI, NI) **then**

propagate perturbation using elaborated corresponding rules (IV.B.3)

**Step 4**

**If**  $j \in O$  **then**  
 $Sum_j = \Delta t'_{j/ij}$   
 $O = O - \{j\}$

**Else**

$i = i + 1$   
 Go to Step 2

**If**  $O = \emptyset$ , **then** ( $O$ : the index of all system output machines)

$\Delta T = (\max_{o \in O} \{d_{ofm} + \Delta t_o\} - T)$

**Endif**

**Endif**

**Step 5**

$(I; J) = (I; J) - (i; j)$

**If**  $(I; J) = \emptyset$  **then** Stop

**Else** Go to Step 1

accumulators are required for the initialization of the gradient computations.

$$A_{ij} = 0 \quad \text{for } i, j = 1, \dots, n. \quad (26)$$

**Step 2** Updates of accumulators

The gradient value is determined at each end of an operation (event), and the gradient of the service time  $t_i$ ,  $\frac{dt_i}{d\theta_i}$ , is added to  $A_{ij}$ . Since in this work  $t_i = \theta_i$  i.e.,  $\frac{dt_i}{d\theta_i} = 1$ ,

$$A_{ii} = A_{ii} + \frac{dt_i}{d\theta_i} \quad (27)$$

If a period NI on machine  $m$  comes to end due to a part moving from machine  $i$  to machine  $m$ ,

$$A_{mj} = A_{ij} \quad j = 1, \dots, n. \quad (28)$$

If a period FO on machine  $i$  comes to end due to a part moving from machine  $i$  to machine  $m$ ,

$$A_{ij} = A_{mj} \quad j = 1, \dots, n. \quad (29)$$

**Step 3** Estimation of  $\partial f / \partial \theta_i$

The value of the accumulator at the simulation's end  $A_{mi}$  is used to calculate an estimate of  $\partial f / \partial \theta_i$

$$\partial f / \partial \theta_i = -(f/T) \cdot A_{mi} \quad (30)$$

**D. STOCHASTIC ALGORITHM**

The developed stochastic algorithm uses the Robbins and Monro optimization technique [37]. The (near-)optimal allocation of processing times and buffer size for the AD system is determined by injecting the gradient estimations from FPA into the stochastic algorithm. The process initiates by selecting the initial values of  $\theta_i$  (buffer sizes and service times) for  $i = 1, \dots, n + m$ . Subsequently, at each iteration  $k$ , the algorithm updates the intermediate variables  $\theta_i$  in the gradient direction, corresponding to the generation of a perturbed path. We use a single run optimization algorithm (i.e., at each end production of 1 units ( $l < L$ ), the update process is activated). The gradients computation method based on FPA is used throughout the configuration update process. Two iterative stochastic algorithms are constructed for the two different spaces (buffer sizes and service times).

$$\theta_i^{k+1} = \theta_i^k + \frac{\alpha}{k} (\partial f / \partial \theta_i) - \frac{1}{m} \sum_{i=1}^m \partial f / \partial \theta_i \quad (\text{buffer sizes}) \quad (31)$$

$$\theta_i^{k+1} = \theta_i^k + \frac{\beta}{k} (\partial f / \partial \theta_i) - \frac{1}{n} \sum_{i=m}^{n+m} \partial f / \partial \theta_i \quad (\text{service times}) \quad (32)$$

Here,  $\alpha$  (and  $\beta$ ) are random constants utilized to determine the step size for the PR.

If some values of  $\theta_i^{k+1}$  are negative, a precaution is proposed as follows:

$$\theta_r^k = \text{Arg min } \theta_i^k; \quad \theta_r^{k+1} = \text{Arg min } \theta_i^{k+1};$$

$$a \sim U(0, 1); h = \left| \frac{a \cdot \theta_r^k}{\theta_r^{k+1} - \theta_r^k} \right| \quad (33)$$

$$\theta_i^{k+1} = \theta_i^k + h \cdot \frac{\alpha}{k} \cdot \left( \frac{\partial f}{\partial \theta_i} - \frac{1}{m} \sum_{i=1}^m \frac{\partial f}{\partial \theta_i} \right) \quad (\text{buffer sizes}) \quad (34)$$

$$\theta_i^{k+1} = \theta_i^k + h \cdot \frac{\beta}{k} \cdot \left( \frac{\partial f}{\partial \theta_i} - \frac{1}{n} \sum_{i=m+1}^{n+m} \frac{\partial f}{\partial \theta_i} \right) \quad (\text{service times}) \quad (35)$$

where, Arg min is an Arena operator which considers the minimum value and  $U(0, 1)$  is a uniform distribution that randomly generates an integer in the interval (0, 1).

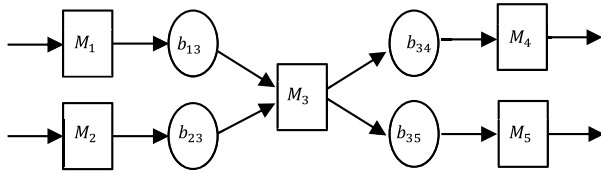


FIGURE 8. Structure of an AD system (5 machines and 4 buffers).

Upon completing the SA, it is necessary to discretize the buffer size values. This discretization is achieved using the Anint operator, which takes into account the nearest integer value and is defined as:

$$\theta_i = \text{Anint}(\theta_i^k); \quad i=1, \dots, m. \quad (36)$$

A trade-off is made between the estimated parameter's precision and the duration of the simulation on the order of  $10^{-4}$  (i.e., when the variables' variation within the same iteration becomes negligible).

## V. EXPERIMENTAL RESULTS

This section presents the experimental results, which are divided into five parts. Firstly, we validate our approach and verify the model's accuracy by examining a homogeneous system where all machines are identical and reliable. Secondly, we conduct experiments to compare the performance of our proposed method with benchmark problems, providing a comprehensive evaluation of its effectiveness. Thirdly, we compare the performance and effectiveness of three different approaches: GA, FPA, and GA-FPA, to assess their contributions. Fourthly, we evaluate the advantages of simultaneously optimizing buffer sizes and service times compared to optimizing buffer capacity alone, highlighting the benefits the former offers. Finally, we perform experiments on a larger AD system (as depicted in Fig. 1) to analyze the performance of our approach in a more complex and realistic setting. We use Arena V14.0 to develop the models and Java to implement the algorithms. Experiments are run on a PC with 8 GB of RAM (intel Core (TM) i5 CPU @ 1.9-GHz). GA uses a population size of 30, tournament selection, and arithmetic crossover operators. The algorithm terminates when it either simulates the predetermined generation number (i.e., 20) or achieves a production rate better than the best production rate attained so far.

### A. ALGORITHMS VALIDATION

To validate our approach and ensure the model's accuracy, we examine a balanced scenario where all machines are identical and reliable. This choice of a homogeneous system allows us to leverage the symmetry in both parameters and structure. This facilitates an intuitive verification of the numerical results and enables us to reach a steady state (if it exists) quickly. In fact, the homogeneity of an assembly (or disassembly) cell implies the convergence towards a state where the buffer sizes of the various stages (upstream or downstream) are similar. This simplifies the analysis and

interpretation of the results. In the specific case illustrated in Fig. 8, we assume that the maximum PR is achieved when the storage spaces possess similar sizes, and all machines have equivalent service times due to the system's symmetry. Assuming that the optimum is not known, GA-FPA is used to estimate the optimum under the constraints:  $B_{max} = 20$  and  $T_{max} = 15$ . We know a priori that the optimum will occur for the configurations  $\{b_{13}, b_{23}, b_{34}, b_{35}\} = \{5, 5, 5, 5\}$  and  $\{t_1, t_2, t_3, t_4, t_5\} = \{3, 3, 3, 3, 3\}$ . The simulation model is executed for 20 runs, producing a total of 10,000 parts.

As expected, our algorithm successfully identifies the optimal solution corresponding to the uniform configuration of the buffer sizes  $\{5, 5, 5, 5\}$ . For the allocation of service times found  $\{2.99998, 3.00001, 3.00000, 3.00001, 3.00000\}$ , there is a slight deviation from the optimal allocation where the processing times of all machines are equal. GA-FPA finds a PR of 0.666420 which is very close to the PR of the optimal configuration (0.666422), with a difference of only 0.0003%. This example demonstrates the ability of the algorithm to converge towards the expected optimal solution for the buffer sizes and towards a solution very close to the optimal solution for the service times, starting from a set of randomly generated initial solutions.

### B. RESULTS ON BENCHMARK PROBLEMS

In this subsection, we assess the performance of GA-FPA compared to the method proposed by [38]. The authors combine a GA and a decomposition-type approximation to simultaneously address the machine selection and buffer allocation problems for AD manufacturing network design to maximize the PR. To evaluate both methods, we employ two test cases, denoted as Instances 1 and 2. The objective of the optimization problem is to maximize the PR while ensuring adherence to a total cost constraint. Notably, Nahas et al. [38] utilize a method that incorporates machine types and buffers as decision variables, while GA-FPA utilizes buffer sizes and service times as decision variables. By considering these differing sets of decision variables, we can effectively compare the performance of the two methods.

#### 1) DESCRIPTION OF INSTANCES

**Instance 1.** Fig. 9 illustrates an AD system comprising 7 machines and 6 buffers. The characteristics of each machine for four different versions ( $V_1$  to  $V_4$ ) are presented in Table 1 (i.e., each machine  $M_i$  ( $i = 1$  to  $7$ ) of version  $V_j$  ( $j = 1$  to  $4$ ) is characterized by its failure rate  $p_i$  and its repair rate  $r_i$ ). The buffer budget ranges from 70 to 90. In the case of the method proposed by Nahas et al. [38], the AD systems are homogeneous, and all processing times are uniform and equal to 1.

**Instance 2.** In this instance, the scenario involves an AD system comprising 7 machines and 6 buffers, illustrated in Fig. 10. This configuration is analogous to Instance 1, although with differences in the arrangement of machines and buffer locations. The characteristics of the machines in this

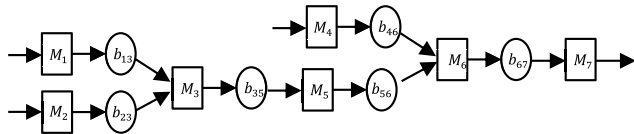


FIGURE 9. Structure of the AD system of instance 1 [38].

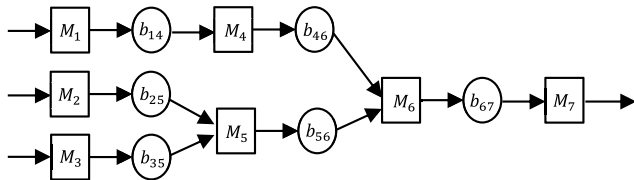


FIGURE 10. Structure of the AD system of instance 2 [38].

TABLE 1. Parameters of machines (4 versions).

Machine	Version	Version			
		V <sub>1</sub>	V <sub>2</sub>	V <sub>3</sub>	V <sub>4</sub>
M <sub>1</sub>	p <sub>1</sub>	0.1000	0.0842	0.0699	0.0616
	r <sub>1</sub>	0.3870	0.3810	0.3540	0.3840
M <sub>2</sub>	p <sub>2</sub>	0.0878	0.0634	0.0536	0.0464
	r <sub>2</sub>	0.3848	0.3735	0.3465	0.3525
M <sub>3</sub>	p <sub>3</sub>	0.1210	0.1000	0.0860	0.0544
	r <sub>3</sub>	0.4028	0.3705	0.3435	0.3525
M <sub>4</sub>	p <sub>4</sub>	0.1460	0.1000	0.0543	0.0443
	r <sub>4</sub>	0.3668	0.4013	0.3398	0.3413
M <sub>5</sub>	p <sub>5</sub>	0.0939	0.1040	0.0920	0.0548
	r <sub>5</sub>	0.3622	0.4118	0.3780	0.3458
M <sub>6</sub>	p <sub>6</sub>	0.1500	0.1000	0.0524	0.0441
	r <sub>6</sub>	0.3638	0.4050	0.3450	0.3720
M <sub>7</sub>	p <sub>7</sub>	0.0700	0.0539	0.0439	0.0400
	r <sub>7</sub>	0.3592	0.3383	0.3532	0.3510

instance remain consistent with those outlined in Instance 1 (refer to Table 1). However, there is a difference in the buffer budget available, ranging from 70 to 90. Similar to Instance 1, for the methodology proposed by Nahas et al. [38], the AD systems maintain homogeneity, with all processing times being uniform and set at 1.

## 2) DISCUSSION OF RESULTS

The aim of these two instances is twofold: firstly, to evaluate the impact of simultaneously optimizing buffer sizes and service rates, and secondly, to provide a benchmark for comparing the simultaneous optimization of buffer sizes and machine selection. To accomplish this, our method employs the optimal values obtained from Nahas et al. [38] through ten runs. Specifically, we utilize the total buffer capacity  $B_{max}$  and the machine versions, as presented in columns 1 and 4 of Table 2. For Instance 1, when the available buffer budget is 70 (resp. 80 and 90),  $B_{max}$  is set to 18 (resp. 28 and 30), while for Instance 2,  $B_{max}$  is set to 21 (resp. 27 and 29). Additionally, the total service time available,  $T_{max}$ , is set to 7.

The results of the two methods are depicted in Table 2. Columns 2 and 3 present the average PR and buffer allocation from Nahas et al.’s method, respectively. On the other hand, columns 5 to 7 display GA-FPA’s average PR, buffer allocation, and service time allocations. An examination of the results from both instances underscores the superior performance of the GA-FPA method in comparison to the simultaneous optimization approach proposed by Nahas et al. [38]. Indeed, upon evaluating the outcomes of our algorithm presented in column 5 alongside the results of Nahas et al.’s work presented in column 2, it is evident that GA-FPA consistently produces a superior average PR. This observation is consistent across different available buffer budgets (70, 80, and 90), showcasing the robustness of GA-FPA. The simultaneous optimization of buffer sizes and service times emerges as a strategic advantage, emphasizing the method’s efficacy in enhancing the average PR of the AD system.

The analysis of buffer size allocations shows that the patterns obtained through the GA-FPA method generally align with those proposed by [38]. However, there are noteworthy distinctions that deserve detailed consideration. For instance, when  $B_{max} = 30$  in Instance 1, Nahas et al. [38] obtain a uniform buffer size allocation of  $\{5,5,5,5,5,5\}$ , differing from GA-FPA’s results of  $\{4,4,8,5,5,4\}$ . This difference is significant as it reflects a more nuanced approach in GA-FPA, particularly in the downstream buffer sizes for the input machines (i.e.,  $M_1$ ,  $M_2$  and  $M_4$ ). Our approach allocates smaller downstream buffer sizes for these input machines, aiming to prevent potential blockages and enhance the overall fluidity of the system.

Additionally, the assembly machine  $M_3$  exhibits a larger downstream stock value (i.e.,  $b_{35}$ ) in GA-FPA’s results. This is designed to accommodate parts from both  $M_1$  and  $M_2$ , and this larger downstream stock can help reduce the blocking periods of these machines and the starvation periods of the downstream machine  $M_5$ . The low service time  $t_3$  of  $M_3$  further supports this claim, as faster execution times necessitate larger downstream buffer sizes for optimal system fluidity. Additionally, the downstream stock of the assembly machine  $M_6$  (i.e.,  $b_{67}$ ) is observed to be smaller than that of  $M_3$  (i.e.,  $b_{35}$ ) in GA-FPA. This allocation could be explained by the fact that the output machine  $M_7$  has an unlimited downstream buffer capacity, ensuring it remains unblocked.

In the second instance, similar patterns are observed. Specifically, the input machines (i.e.,  $M_1$ ,  $M_2$  and  $M_3$ ) exhibit smaller downstream buffer sizes (i.e.,  $b_{14}$ ,  $b_{25}$  and  $b_{35}$ ). This allocation is aimed at mitigating potential blockages and enhancing the overall flow within the system. Additionally, the assembly machine  $M_5$  stands out with a substantial downstream stock (i.e.,  $b_{56}$ ) and a lower service time  $t_5$ . This configuration aligns with the notion that maintaining a larger downstream stock can effectively reduce periods of starvation for the subsequent machine,  $M_6$ . This effect is particularly pronounced in scenarios with rapid execution times, where optimal system fluidity necessitates larger downstream buffer

TABLE 2. Results for different buffer spaces.

Case	Nahas et al. [38]			GA-FPA		
	PR	Buffer	Version	PR	Buffer	Service
<b>Instance 1</b>		$\{b_{13}, b_{23}, b_{35}, b_{46}, b_{56}, b_{67}\}$			$\{b_{13}, b_{23}, b_{35}, b_{46}, b_{56}, b_{67}\}$	$\{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$
$B_{max} = 18$	0.60330	{3,3,5,2,4,1}	{1,1,1,2,1,3,1}	0.60968	{2,3,5,3,3,2}	{1.00,0.99,0.99,1.04,0.96,1.00,1.02}
$B_{max} = 28$	0.64379	{5,5,5,4,5,4}	{1,1,1,2,1,3,1}	0.65116	{4,4,7,4,5,4}	{1.02,1.01,0.96,0.98,0.97,1.01,1.05}
$B_{max} = 30$	0.65903	{5,5,5,5,5,5}	{1,1,3,2,2,3,1}	0.66444	{4,4,8,5,5,4}	{0.99,1.00,0.99,1.01,0.99,1.02,1.00}
<b>Instance 2</b>		$\{b_{14}, b_{25}, b_{35}, b_{46}, b_{56}, b_{67}\}$			$\{b_{14}, b_{25}, b_{35}, b_{46}, b_{56}, b_{67}\}$	$\{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$
$B_{max} = 21$	0.60352	{2,3,4,5,5,2}	{1,1,1,2,1,2,1}	0.62087	{2,3,4,3,5,4}	{1.03,0.99,0.98,1.00,0.98,0.98,1.04}
$B_{max} = 27$	0.64517	{4,5,5,5,5,3}	{1,1,1,2,1,3,1}	0.65804	{4,4,5,4,6,4}	{1.04,1.01,0.98,0.98,0.97,1.01,1.01}
$B_{max} = 29$	0.66877	{5,5,5,5,5,4}	{1,1,1,2,4,3,1}	0.67764	{4,4,5,4,7,5}	{1.00,1.01,0.97,1.03,0.97,0.97,1.03}

sizes. Interestingly, the assembly machine  $M_6$  has a smaller downstream stock (i.e.,  $b_{67}$ ) compared to assembly machine  $M_5$  (i.e.,  $b_{56}$ ), which may be explained by considering that the output machine  $M_7$  possesses an infinite downstream buffer capacity, preventing it against blockages.

To summarize, Nahas et al. [38] propose that the selection of appropriate buffer sizes and machine types plays a pivotal role in enhancing the PR of AD systems. Moreover, the simultaneous optimization of service times and buffer sizes can yield even more favorable outcomes. These allocation strategies offer valuable insights to decision-makers seeking to optimize the efficiency of their manufacturing systems.

C. COMPARISON BETWEEN GA, PA AND GA-FPA

In the experimental setup outlined in Subsections V-B, V-C, and V-D, our primary objective was to assess the average PR through the simulation of the production of 10,000 parts across 20 replications. The total processing time is set to be  $(3 \cdot n)$ -time unit, where  $n$  represents the number of machines. All machines are identical and are susceptible to failures. These failures and repair processes are geometrically distributed with an average of MTBF=70 and MTTR=10 respectively. The focus of our assessment is on the hybrid optimization algorithm GA-FPA, and we compare its performance against the individual algorithms GA and FPA.

The results, graphically presented in Fig. 11, depict the evolution of the average PR over 20 generations. The results show that GA-FPA exhibits superior performance compared to GA and FPA when considered individually, achieving the best solution with a PR value of 0.5118, outperforming GA (0.4979) and FPA (0.4648). A key advantage of GA-FPA lies in its capacity to quickly identify promising regions within the search space. This capability is attributed to GA’s exploration ability, allowing GA-FPA to discern interesting areas within

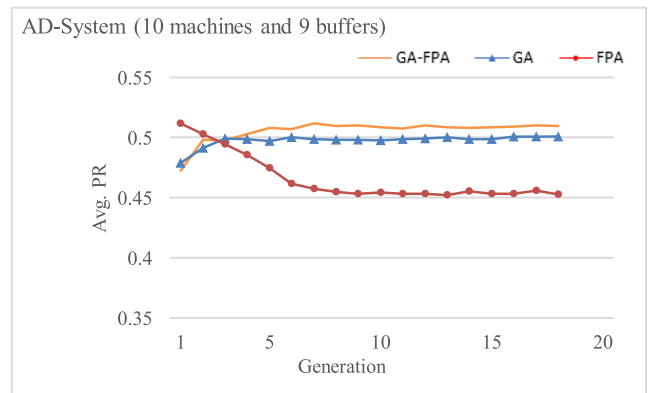


FIGURE 11. Convergence curves of GA-FPA, GA, and FPA.

a mere four generations, as represented in Fig. 11. Following this initial exploration, the algorithm strategically intensifies its search within these identified regions, leveraging the exploitation ability of FPA. This combination proves particularly advantageous in the context of complex, large-scale combinatorial optimization models, where traditional methods are known for their computationally intensive nature.

D. COMPARISON BETWEEN SIMULTANEOUS ALLOCATION OF BUFFERS AND SERVICE TIMES AND BUFFER SIZES ALLOCATION

This subsection investigates the comparative evaluation of optimizing buffer sizes and processing times simultaneously versus the optimization of buffer size. The experiment considers an AD system comprising ten machines and nine buffers. In cases where buffer size optimization is performed, the total service time is held constant at  $T_{max} = 30$ . The maximum buffer size  $B_{max}$  is varied across five cases: 45, 180, 270, 360,

**TABLE 3.** Results of buffer, and simultaneous allocations (AD system with 10 machines and 9 buffers).

$B_{max}$	Avg. PR		CPU (s)	
	Simultaneous allocation	Buffer allocation	Simultaneous allocation	Buffer allocation
45	<b>0.48033</b>	0.47475	<b>1434</b>	1675
180	<b>0.54467</b>	0.54453	<b>1837</b>	2091
270	<b>0.55356</b>	0.55092	<b>1909</b>	2089
360	<b>0.55678</b>	0.55219	<b>2039</b>	2899
450	<b>0.55926</b>	0.55902	<b>2353</b>	2525

**TABLE 4.** Results for AD system (10 machines and 9 buffers).

$B_{max}$	Optimal buffer sizes $b^*$									Optimal service times $t^*$									Avg. PR	CPU(s)	
45	5	5	6	6	6	6	4	4	3	3.00	2.99	2.99	3.00	3.03	3.00	2.98	2.99	3.00	3.02	0.48033	1434
90	9	10	10	12	9	11	11	9	9	2.97	2.99	2.99	2.99	2.99	3.00	3.01	2.98	3.06	3.02	0.51181	1478
135	15	16	18	15	14	15	15	15	12	3.00	2.99	2.99	3.00	3.02	2.98	2.99	3.00	3.01	3.02	0.53229	1337
180	20	20	22	19	20	19	21	20	19	3.00	3.01	2.99	3.00	2.98	3.00	3.00	2.99	3.00	3.01	0.54467	1837
225	26	26	25	27	25	24	23	25	24	3.00	3.01	2.97	2.99	2.99	2.99	2.98	3.01	3.03	3.03	0.54950	1762
270	37	36	33	35	23	27	28	26	25	3.01	3.03	2.97	2.99	3.00	2.99	3.01	3.01	2.97	3.02	0.55356	1909
315	34	33	37	36	39	36	34	35	31	2.97	3.03	2.97	3.02	2.99	3.02	3.01	2.99	3.00	3.00	0.55557	2181
360	42	43	47	34	46	35	44	34	35	3.01	2.98	3.02	2.98	3.04	2.98	2.97	2.99	3.00	3.03	0.55678	2039
405	44	46	42	48	47	46	47	46	39	3.00	2.99	2.97	2.99	3.03	3.00	3.01	3.00	2.99	3.02	0.55810	1516
450	52	50	51	49	52	50	48	50	48	3.00	2.98	2.99	2.99	3.02	3.00	3.00	3.01	3.01	3.00	0.55926	2353

and 450. The results of the experiment are detailed in Table 3. Column 1 displays the maximum buffer size  $B_{max}$  for each case, while columns 2 and 3 present the average PR obtained from simultaneous allocation and buffer allocation alone, respectively. The last two columns provide the computation time required for each scenario, measured in seconds.

The experimental results demonstrate that the simultaneous optimization of buffer sizes and service times leads to superior average PR results compared to optimizing buffer size alone. Additionally, the simultaneous allocation is faster than buffer allocation alone in computation time. Therefore, our results provide a clear advantage that simultaneously optimizing buffer sizes and service times in AD systems has a clear advantage over-optimizing buffer size alone. This approach results in significant performance improvements, making it a promising method for optimizing AD system design and planning. Moreover, these results motivate further exploration and development of other meta-heuristic-based approaches for the BSTAP.

**E. RESULTS ON LARGER PROBLEMS**

In this subsection, our objective is to assess the efficacy of the GA-FPA hybrid algorithm in addressing the complex challenge of jointly optimizing buffer sizes and processing times for AD systems, characterized by a substantially larger

solution space. The AD system under consideration, illustrated in Fig. 1, consists of ten machines and nine buffers. Notably, the total buffer size is intentionally chosen as a multiple of the number of buffers, specifically a multiple of 9. This deliberate choice in the configuration is designed to observe the buffer size allocation’s behavior relative to a uniform allocation of buffer sizes.

Table 4 presents the results obtained through the application of GA-FPA. The first column enumerates the values of  $B_{max}$ , ranging from 45 to 450 with increments of 45. The second and third columns show the optimal allocations of buffer sizes ( $b^*$ ) and processing times ( $t^*$ ), respectively. The fourth column provides the average PR achieved through the optimal allocation. Lastly, the last column presents the computation times associated with each scenario, expressed in seconds. Through this instance, we aim to assess the performance of GA-FPA in addressing the intricate challenge of simultaneous service time and buffer size allocation for larger AD systems, where the solution space is notably expanded.

The results presented in Table 4 highlight the performance of the GA-FPA hybrid algorithm, consistently producing optimal solutions across twenty runs within a reasonable computation time, as indicated in Column 5. Notably, even in the case with the longest computation time, averaging 39.2 minutes for a maximum buffer capacity  $B_{max}$  of 450, the algorithm’s efficiency remains within acceptable bounds,



particularly when benchmarked against industry standards (Respen et al., [39]).

Several noteworthy observations emerge from the analysis of the experiments conducted on the joint allocation of buffer capacity and service time:

1. **Symmetry in Buffer Sizes:** The optimal configurations frequently exhibit symmetry between the stages containing input machines  $M_1$  and  $M_2$ , as illustrated in Fig. 1. This symmetry is intuitive, considering that there is no inherent difference between the machines supplying buffers  $b_{13}$  and  $b_{23}$ .
2. **Downstream Buffer of Output Machine  $M_{10}$ :** The upstream buffer  $b_{910}$  of the output machine  $M_{10}$  is found to require a smaller capacity. This is attributed to the infinite downstream storage capacity of output machine  $M_{10}$ , enabling sustained operation without the need for additional buffer space. Consequently, allocating additional space to a stock that tends to deplete more rapidly is deemed unnecessary.
3. **Uniform Allocation:** Despite the stochastic nature of the environment, there persists an almost uniform allocation of buffer sizes. This uniformity can be attributed to the inherent symmetry of the system, characterized by identical machines. The algorithm consistently converges to an optimal and symmetrical result, even in the presence of stochastic variations.

## VI. CONCLUSION

This paper presents an algorithm that expands the classical BAP design to concurrently allocate service times and buffer sizes in AD systems with unreliable machines. The algorithm involves determining the service time of each machine and the size of each buffer, ensuring that the whole buffer storage and total processing time for the system do not exceed predefined limits. We propose a hybrid approach named GA-FPA, which integrates GA with FPA. The key element of this method is a stochastic algorithm that estimates production rate gradients, with GA providing initial solutions. The GA's exploration and diversification capabilities enable rapid access to (near-) optimal solutions, while FPA's intensification and exploitation abilities thoroughly explore these solution spaces to identify improved solutions. One notable benefit of the presented approach lies in its capacity to provide a solution to the design problem through only one long simulation. This approach meaningfully reduces the overall computation time required.

The investigation of the impact of the joint simultaneous allocations (i.e., buffer capacity/service time and buffer capacity/type of machine) compared to the allocation of buffer capacity alone on the evolution of the production rate shows that the simultaneous allocation, whether it is buffer size with service time or with machine type, improves the production rate. Furthermore, our proposed method of simultaneous allocation of service times and buffer sizes showed an improved production rate compared to the method of simultaneous allocation of machine types and buffer sizes [38].

Experiments on a larger problem (10 machines and 9 buffers) demonstrate the performance of the proposed method in terms of solution quality and computation times. GA-FPA outperforms both FPA and GA algorithms when considered alone for optimization problems. Future research could explore the limitations of GA-FPA, especially concerning solution quality and computation time, and assess its effectiveness with a higher number of machines (up to 40 or 50) common in industrial environments. Additionally, investigating the proposed approach for multi-objective problems, such as minimizing total buffer size or work-in-process (WIP) inventory, could be valuable. Another potential avenue is the simultaneous design of an AD system considering all three decision variables: buffer sizes, machine types, and service times, requiring an integrated optimization approach to address their interdependencies and enhance overall system performance.

## DATA AVAILABILITY STATEMENT

The paper includes no data.

## DISCLOSURE STATEMENT

The authors report there are no competing interests to declare.

## REFERENCES

- [1] S. Gershwin, *Manufacturing Systems Analysis*, vol. 7632. Engle-Wood Cliffs, NJ, USA: Prentice-Hall, 1994, pp. 69–93.
- [2] C. G. S. Sikora, *Assembly-Line Balancing Under Demand Uncertainty* (Gabler Theses). Wiesbaden, Germany: Springer, 2022, doi: 10.1007/978-3-658-36282-9.
- [3] J.-S. Tancrez, "A decomposition method for assembly/disassembly systems with blocking and general distributions," *Flexible Services Manuf. J.*, vol. 32, no. 2, pp. 272–296, Jun. 2020, doi: 10.1007/s10696-019-09332-z.
- [4] N. Cheikhrouhou and B. Descotes-Genon, "Buffer design in stochastic assembly/disassembly systems," *IFAC Proc. Volumes*, vol. 33, no. 17, pp. 867–872, Jul. 2000.
- [5] R. Arista, X. Zheng, J. Lu, and F. Mas, "An ontology-based engineering system to support aircraft manufacturing system design," *J. Manuf. Syst.*, vol. 68, pp. 270–288, Jun. 2023, doi: 10.1016/j.jmsy.2023.02.012.
- [6] S. Lidberg, T. Aslam, L. Pehrsson, and A. H. C. Ng, "Optimizing real-world factory flows using aggregated discrete event simulation modelling," *Flexible Services Manuf. J.*, vol. 32, no. 4, pp. 888–912, Dec. 2020, doi: 10.1007/s10696-019-09362-7.
- [7] M. Chica, J. Bautista, and J. de Armas, "Benefits of robust multiobjective optimization for flexible automotive assembly line balancing," *Flexible Services Manuf. J.*, vol. 31, no. 1, pp. 75–103, Mar. 2019, doi: 10.1007/s10696-018-9309-y.
- [8] Y.-C. L. Ho and X.-R. Cao, *Perturbation Analysis of Discrete Event Dynamic Systems*. Germany: Springer, 2012.
- [9] S. Scrivano, B. Tan, and T. Tolio, "Continuous-flow simulation of manufacturing systems with assembly/disassembly machines, multiple loops and general layout," *J. Manuf. Syst.*, vol. 69, pp. 103–118, Aug. 2023, doi: 10.1016/j.jmsy.2023.05.028.
- [10] M. Hanafy and H. ElMaraghy, "Modular product platform configuration and co-planning of assembly lines using assembly and disassembly," *J. Manuf. Syst.*, vol. 42, pp. 289–305, Jan. 2017, doi: 10.1016/j.jmsy.2016.12.002.
- [11] S. Xi, J. M. Smith, Q. Chen, N. Mao, H. Zhang, and A. Yu, "Simultaneous machine selection and buffer allocation in large unbalanced series-parallel production lines," *Int. J. Prod. Res.*, vol. 60, no. 7, pp. 2103–2125, Apr. 2022.
- [12] S. G. Powell and D. F. Pyke, "Buffering unbalanced assembly systems," *IIE Trans.*, vol. 30, no. 1, pp. 55–65, 1997, doi: 10.1023/a:1007493512502.
- [13] L. Fuxman, "Optimal buffer allocation in asynchronous cyclic mixed-model assembly lines," *Prod. Oper. Manage.*, vol. 7, no. 3, pp. 294–311, Sep. 1998, doi: 10.1111/j.1937-5956.1998.tb00458.x.

- [14] T. Yamada and M. Matsui, "A management design approach to assembly line systems," *Int. J. Prod. Econ.*, vol. 84, no. 2, pp. 193–204, May 2003.
- [15] N. Hemachandra and S. K. Eedupuganti, "Performance analysis and buffer allocations in some open assembly systems," *Comput. Oper. Res.*, vol. 30, no. 5, pp. 695–704, Apr. 2003.
- [16] S. Shaaban, T. McNamara, and V. Dmitriev, "Asymmetrical buffer allocation in unpaced merging assembly lines," *Comput. Ind. Eng.*, vol. 109, pp. 211–220, Jul. 2017, doi: [10.1016/j.cie.2017.05.008](https://doi.org/10.1016/j.cie.2017.05.008).
- [17] A. A. Bulgak, "Analysis and design of split and merge unpaced assembly systems by metamodelling and stochastic search," *Int. J. Prod. Res.*, vol. 44, nos. 18–19, pp. 4067–4080, Sep. 2006, doi: [10.1080/00207540600564625](https://doi.org/10.1080/00207540600564625).
- [18] H. Chehade, F. Yalaoui, L. Amodeo, and F. Dugardin, "Buffers sizing in assembly lines using a Lorenz multiobjective ant colony optimization algorithm," in *Proc. Int. Conf. Mach. Web Intell.*, Oct. 2010, pp. 283–287.
- [19] L. Bertazzi, "Determining the optimal dimension of a work-in-process storage area," *Int. J. Prod. Econ.*, vol. 131, no. 2, pp. 483–489, Jun. 2011.
- [20] A. Alfieri, A. Matta, and E. Pastore, "The time buffer approximated buffer allocation problem: A row-column generation approach," *Comput. Oper. Res.*, vol. 115, Mar. 2020, Art. no. 104835, doi: [10.1016/j.cor.2019.104835](https://doi.org/10.1016/j.cor.2019.104835).
- [21] H. Tempelmeier, "Practical considerations in the optimization of flow production systems," *Int. J. Prod. Res.*, vol. 41, no. 1, pp. 149–170, Jan. 2003, doi: [10.1080/00207540210161641](https://doi.org/10.1080/00207540210161641).
- [22] J. M. Smith, "Simultaneous buffer and service rate allocation in open finite queueing networks," *IIE Trans.*, vol. 50, no. 3, pp. 203–216, Mar. 2018, doi: [10.1080/24725854.2017.1300359](https://doi.org/10.1080/24725854.2017.1300359).
- [23] M. S. Hillier and F. S. Hillier, "Simultaneous optimization of work and buffer space in unpaced production lines with random processing times," *IIE Trans.*, vol. 38, no. 1, pp. 39–51, Jan. 2006, doi: [10.1080/07408170500208289](https://doi.org/10.1080/07408170500208289).
- [24] S. Spieckermann, K. Gutenschwager, H. Heinzl, and S. Voß, "Simulation-based optimization in the automotive industry—A case study on body shop design," *Simulation*, vol. 75, nos. 5–6, pp. 276–286, 2000.
- [25] F. R. B. Cruz, G. Kendall, L. While, A. R. Duarte, and N. L. C. Brito, "Throughput maximization of queueing networks with simultaneous minimization of service rates and buffers," *Math. Problems Eng.*, vol. 2012, Mar. 2012, Art. no. e692593, doi: [10.1155/2012/692593](https://doi.org/10.1155/2012/692593).
- [26] F. R. B. Cruz, A. R. Duarte, and G. L. Souza, "Multi-objective performance improvements of general finite single-server queueing networks," *J. Heuristics*, vol. 24, no. 5, pp. 757–781, Oct. 2018, doi: [10.1007/s10732-018-9379-8](https://doi.org/10.1007/s10732-018-9379-8).
- [27] K. Kassoul, N. Cheikhrouhou, and N. Zufferey, "Simultaneous allocation of buffer capacities and service times in unreliable production lines," *Int. J. Prod. Res.*, pp. 1–21, Jan. 2023, doi: [10.1080/00207543.2023.2168310](https://doi.org/10.1080/00207543.2023.2168310).
- [28] M. D. Mascolo, R. David, and Y. Dallery, "Modeling and analysis of assembly systems with unreliable machines and finite buffers," *IIE Trans.*, vol. 23, no. 4, pp. 315–330, Dec. 1991, doi: [10.1080/07408179108963866](https://doi.org/10.1080/07408179108963866).
- [29] T. Harada and E. Alba, "Parallel genetic algorithms: A useful survey," *ACM Comput. Surv.*, vol. 53, no. 4, pp. 1–39, Jul. 2021, doi: [10.1145/3400031](https://doi.org/10.1145/3400031).
- [30] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future," *Multimedia Tools Appl.*, vol. 80, no. 5, pp. 8091–8126, Feb. 2021, doi: [10.1007/s11042-020-10139-6](https://doi.org/10.1007/s11042-020-10139-6).
- [31] K. Kassoul, N. Cheikhrouhou, and N. Zufferey, "Buffer allocation design for unreliable production lines using genetic algorithm and finite perturbation analysis," *Int. J. Prod. Res.*, vol. 60, no. 10, pp. 3001–3017, May 2022, doi: [10.1080/00207543.2021.1909169](https://doi.org/10.1080/00207543.2021.1909169).
- [32] D. Newton, F. Yousefian, and R. Pasupathy, "Stochastic gradient descent: Recent trends," in *Recent Advances in Optimization and Modeling of Contemporary Problems*. Catonsville, MD, USA: INFORMS, 2018, pp. 193–220.
- [33] A. Amirteimoori and R. Kia, "Concurrent scheduling of jobs and AGVs in a flexible job shop system: A parallel hybrid PSO-GA meta-heuristic," *Flexible Services Manuf. J.*, vol. 35, no. 3, pp. 727–753, Sep. 2023, doi: [10.1007/s10696-022-09453-y](https://doi.org/10.1007/s10696-022-09453-y).
- [34] C. Jin, F. Li, E. C. C. Tsang, L. Bulysheva, and M. Y. Kataev, "A new compound arithmetic crossover-based genetic algorithm for constrained optimisation in enterprise systems," *Enterprise Inf. Syst.*, vol. 11, no. 1, pp. 122–136, Jan. 2017, doi: [10.1080/17517575.2015.1080302](https://doi.org/10.1080/17517575.2015.1080302).
- [35] R. Suri and Y. T. Leung, "Single run optimization of a SIMAN model for closed loop flexible assembly systems," in *Proc. 19th Conf. Winter Simulation*, Atlanta, GA, USA, Dec. 1987, pp. 738–748.
- [36] R. Suri and Y. T. Leung, "Single run optimization of a SIMAN model for closed loop flexible assembly systems," in *Proc. 19th Conf. Winter Simul. (WSC)*, 1987, pp. 738–748.
- [37] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Stat.*, vol. 22, no. 3, pp. 400–407, Sep. 1951.
- [38] N. Nahas, M. Nourelfath, and M. Gendreau, "Selecting machines and buffers in unreliable assembly/disassembly manufacturing networks," *Int. J. Prod. Econ.*, vol. 154, pp. 113–126, Aug. 2014, doi: [10.1016/j.iipe.2014.04.011](https://doi.org/10.1016/j.iipe.2014.04.011).
- [39] J. Respen, N. Zufferey, and P. Wieser, "Three-level inventory deployment for a luxury watch company facing various perturbations," *J. Oper. Res. Soc.*, vol. 68, no. 10, pp. 1195–1210, Oct. 2017.



**KHELIL KASSOUL** received the Ph.D. degree in economics and management from Geneva School of Economics and Management (GSEM). From 2003 to 2021, he held the position of a Teacher in mathematics and physics for Swiss maturity and Baccalaureat classes. He is currently a Researcher with Geneva School of Business Administration (HEG), University of Applied Sciences of Western Switzerland (HES-SO). Previously, he was an Associate Member with European Organization for Nuclear Research (CERN). His research interests include applied mathematics, production systems modeling, simulation and optimization, inventory management, and artificial intelligence.



**NAOUFEL CHEIKHROUHOU** (Senior Member, IEEE) received the Graduate degree from the School of Engineers of Tunis and the Ph.D. degree in industrial engineering from the Grenoble Institute of Technology, in 2001. He is currently a Professor in supply chain, logistics and operations management with Geneva School of Business Administration, University of Applied Sciences Western Switzerland. Previously, he was leading the Operations Management Research Group, Swiss Federal Institute of Technology at Lausanne (EPFL). His main research interests include modeling, simulation and optimization of enterprise networks, behavioral operations management, and demand planning and forecasting. He received the Burbidge Award, in 2003, and the BG Ingénieurs Conseils Award, in 2013, for his contribution to the research excellence in the fields. Leading different projects with the collaboration of national and international industrial companies, he has published more than 100 papers in scientific journals and conferences and regularly acts as a keynote speaker in academic and industrial international conferences.

...