Adaptive Dynamic Jumping Particle Swarm Optimization for Buffer Allocation in Unreliable Production Lines

Kassoul, Khelil; Cheikhrouhou, Naoufel; Zufferey, Nicolas

How to cite

KASSOUL, Khelil, CHEIKHROUHOU, Naoufel, ZUFFEREY, Nicolas. Adaptive Dynamic Jumping Particle Swarm Optimization for Buffer Allocation in Unreliable Production Lines. In: IEEE access, 2023, vol. 11, p. 90410–90420. doi: 10.1109/ACCESS.2023.3307017

This publication URL: https://archive-ouverte.unige.ch/unige:171810
Publication DOI: 10.1109/ACCESS.2023.3307017

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.Doi Number

Adaptive Dynamic Jumping Particle Swarm Optimization for Buffer Allocation in Unreliable Production Lines

Khelil Kassoul [1], Naoufel Cheikhrouhou[1,3], Senior Member, IEEE and, Nicolas Zufferey[2]

[1]Geneva School of Business Administration, University of Applied Sciences Western Switzerland, HES-SO, 1227 Geneva, Switzerland
[2]Geneva School of Economics and Management, GSEM, University of Geneva, 1211 Geneva 4, Switzerland
[3]IFM Business School, 1205 Geneva, Switzerland

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

ABSTRACT Over the past half-century, the Buffer Allocation Problem (BAP) in production lines has remained a topic of continuous interest and research. In this context, the BAP refers to determining the optimal allocation of buffers along a production line to maximize efficiency and productivity. This paper presents a novel approach to address the BAP in unreliable serial production lines. The objective is to maximize the production rate of the production line, which directly influences its overall performance and profitability. The proposed approach introduces an adaptive simulation-optimization methodology, APSO, that leverages the particle swarm optimization (PSO) algorithm. PSO is a metaheuristic optimization technique inspired by the behavior of bird flocking or fish schooling, where particles explore the solution space to find optimal solutions. The novelty of this approach lies in integrating a jumping strategy into the PSO algorithm's velocity equation. The jumping strategy incorporates logarithmic and exponential functions into the velocity equation of the PSO algorithm, utilizing dynamic parameters. This modification enables the algorithm to quickly converge towards (or very close to) optimal solutions. By incorporating this jumping strategy, the proposed approach enhances the algorithm's exploration-exploitation balance, efficiently navigating complex solution spaces and overcoming local optima. To evaluate the effectiveness of the proposed method, extensive numerical experiments are conducted using various instances of production lines, ranging from 3 to 100 machines. Additionally, benchmark algorithms from the existing literature are employed for comparison purposes. The obtained results from these experiments serve as empirical evidence to demonstrate the efficiency and accuracy of the proposed approach. The results indicate that the proposed adaptive approach outperforms the benchmark algorithms regarding efficiency and solution quality.

INDEX TERMS Buffer allocation; particle swarm optimization; production rate; simulation; unreliable serial production lines

I. INTRODUCTION

This Manufacturing is critical to the global economy and prosperity [1]. In recent decades, a significant body of literature has studied serial production lines extensively utilized in manufacturing systems. These production lines comprise a series of machines arranged sequentially, with buffers between adjacent machines. Units or items, such as materials, parts, or products, traverse these machines following predetermined sequences. Numerous studies in the manufacturing field aim to enhance the effectiveness of these production lines, considering various objectives such as production rate and profit. However, the efficiency of the production line can be hindered by stochastic factors, including machine failures and repairs, random service times, and consequently events like starving and blocking. These stochastic disruptions can impede the smooth flow of materials and adversely affect the production line's overall performance. One approach to mitigate the impact of these disruptions is by allocating additional buffer sizes along the production line. This buffer allocation strategy helps increase the line's average production rate (PR) while mitigating the propagation of disruptions. Nevertheless, this solution introduces additional challenges as it may lead to higher work-

in-process (WIP) inventories, increased capital investment costs, and more floor space requirements. Therefore, determining the optimal allocation of buffers becomes a challenge manufacturing optimization problem, aiming to reduce costs while maintaining a desirable PR.

This significant challenge in the design of serial production lines is called the buffer allocation problem (BAP) and has garnered considerable attention in recent literature [2], [3]. The BAP varies depending on the line's topology and the solution methodology employed. These variations include primal/dual problems, work-in-process (WIP) minimization, and constraint/unconstraint profit maximization. Production lines can consist of reliable machines that do not experience failures or unreliable machines that are susceptible to failures. Solving the BAP entails developing algorithms based on evaluative search methods (e.g., Markovian state model, decomposition, simulation) or generative search methods (e.g., metaheuristics like particle swarm optimization, simulated annealing and genetic algorithms, search algorithms, and dynamic programming).

The BAP can exhibit variations in its objective function and constraints, depending on the specific version of the problem being addressed. The primal problem formulation aims to minimize the total buffer capacity needed for the production line while satisfying a constraint related to the desired production rate. Conversely, the dual problem formulation aims to maximize the achievable production rate while working within a specified total buffer capacity. In the context of constraint profit maximization, the objective is to maximize the profit while adhering to a predetermined total buffer capacity. This formulation considers both the buffer capacity constraint and the goal of maximizing profitability. On the other hand, the unconstraint profit maximization formulation relaxes the constraint on buffer capacity, allowing for more flexibility in the pursuit of profit maximization.

The BAP is a well-known combinatorial optimization problem classified as NP-hard [4]. Closed-form formulas that relate decision variables (e.g., buffer capacities) to performance measures are challenging to establish, mainly when the production line comprises more than two machines. Additionally, the computational complexity of the BAP increases significantly as the problem size grows. Effectively and reliably solving the BAP remains challenging, particularly for large-scale production lines.

The paper is structured as follows: Section II presents a thorough literature review of solution approaches for the BAP. The mathematical formulation of the problem is provided in Section III. Section IV outlines the optimization method and the associated assumptions. The results of numerical experiments are discussed in Section V. Finally, the paper concludes with a summary and highlights potential avenues for future research.

II. LITERATURE REVIEW

The solution methods used to address the BAP can be categorized into iterative optimization methods (generative and evaluative methods), integrated methods, and explicit solutions [3]. Explicit solutions involve deriving formulas or rules that describe the allocation of buffers based on given optimization problem parameters. These rules and formulas are obtained through exact analytical methods, analysis of optimal solution characteristics, or approximate performance evaluation techniques [9]. Integrated methods employ performance evaluation formulations based on analytical results or sampling and utilize integer programming and standard linear solvers to solve the optimization problem [10], [11]. Among these approaches, the iterative method is the most widely used, requiring both generative and evaluative tools [12]. The generative method generates solutions, while the evaluative method assesses their performance.

Evaluation methods can be divided into two categories: analytical methods and simulations. Analytical methods, such as Markovian state models, comprehensively characterize system features but are only suitable for small production lines due to their ample state space and computational complexity [13], [14]. Simulation methods, on the other hand, use simulation packages like Arena to evaluate the performance of large and complex production systems [15], [16]. Simulations benefit complex design challenges where other assessment techniques may lack precision [2]. However, simulations can be time-consuming [17].

Three commonly used approximation methods for larger production lines are the generalized expansion method [18], [19], the decomposition method [20]–[22], and the aggregation method [23], [24]. The generalized expansion method utilizes queuing models and applies to serial production lines and split and merge configurations, accommodating reliable machines and random distributed service times. Decomposition methods divide the system into smaller subsystems to reduce computational effort. These methods are applied assuming that service times (or repair/failure rates) follow exponential or deterministic distributions. Aggregation methods, on the other hand, replace two-machine-and-one-buffer components with equivalent single machines, recursively applying this aggregation until the first or last station is reached.

In terms of generative (optimization) methods, extensive research has focused on finding (near-)optimal values for decision variables [25]. Complete Enumeration (CE) is the simplest generative method but is only suitable for small systems. However, due to computational constraints, no approach efficiently solves the BAP for large production lines [26], [27]. To address the combinatorial nature of the BAP and improve computational efficiency, various optimization methods have been proposed, including dynamic programming, search methods, and metaheuristics.

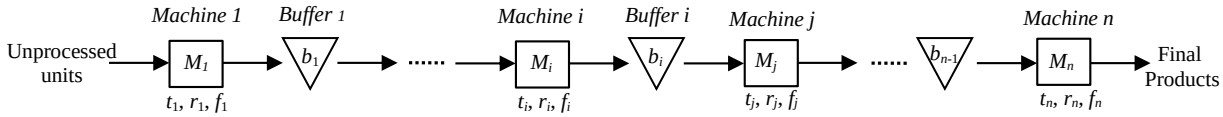


FIGURE 1. **Serial production line.**

Dynamic programming is a powerful optimization method that divides the BAP into subproblems to reduce combinatorial complexity [28], [29]. However, it is typically applied to reliable production lines and makes certain assumptions. Search methods aim to find feasible solutions close to (near-)optimal solutions to overcome the explosion of feasible solutions [2]. Commonly used search methods include the gradient method [30], [31], the degraded ceiling method [32], and Hooke and Jeeves method [33]. However, search methods are prone to get trapped in local optima and struggle to assess the impact of small changes in buffer capacities on overall system performance. To address these issues, heuristic search algorithms have been developed [34], [35]. Metaheuristics are another category of optimization methods that effectively manage the search process and explore the solution space within reasonable computing times. Examples of metaheuristics include genetic algorithms (GA) [36], simulated annealing (SA) [37], tabu search (TS) [38], and ant colony optimization [39]. Hybridization of metaheuristics with other methods has become a trend to enhance their efficiencies, such as combining TS with Nested Partitions [40], GA with SA [41], PSO and Optimal Computing Budget Allocation (OCBA) [42], and GA with Finite Perturbation analysis (FPA) [43].

Regarding PSO, there are only two known papers addressing the BAP [7], [8], both focusing on small production lines (8 machines). Lin and Chiu [42] hybridizes PSO with OCBA but for the Resource Allocation Problem. In this context, we aim to demonstrate that our proposed APSO efficiently addresses the BAP in large unreliable production lines (3 to 100 machines). We highlight the efficient convergence of APSO, achieved through integrating a jumping technique in the velocity equation, employing exponential and logarithmic functions. This enables particles to make leaps and avoid local optima. Determining each particle's role and action set is a crucial task, similar to other optimization paradigms involving "agents" [44].

III. PROBLEM FORMULATION

Serial production lines, also referred to as flow lines, tandem lines, transfer lines, and similar terms, typically consist of multiple machines through which materials (parts) flow from an external source into the system. The material starts at the first machine, denoted as M_1, then proceeds to the first buffer, known as b_1. From there, it moves on to the next machine, M_2, and continues this progression until it reaches the final machine, M_n, where it exits the system. Figure 1 illustrates such a production line, demonstrating the connection between n machines $(M_1, \ldots, M_n)$ and their separation by $(n-1)$ buffers $(b_1, \ldots, b_{n-1})$. Each buffer, b_i, represents the capacity between two adjacent machines, M_i and M_{i+1}. Additionally, we possess information about the failure (f_i) and repair (r_i) probabilities for each machine, along with the service time (t_i) required to process a single unit.

In a serial production line, machines can be either operational (UP) or non-operational (DOWN) due to internal failures. Several assumptions guide our work: the first machine is always supplied with parts, and the last machine is never blocked; machines can only fail when they are UP and not blocked or starved; the repair and failure rates of machines follow a geometric distribution; delays in part supply are negligible; when a machine is UP, it can be starved (or have a Null Input (NI)) if its upstream buffer is empty, and it can be blocked (or have a Full Output (FO)) if its downstream buffer is full.

To quantify machine reliability, we define the repair probability (r_i) as the likelihood of a machine being UP for the next part, given that it was DOWN for the current part. The failure probability (f_i) represents the probability of a machine being DOWN for the next part, given that it was UP for the current part. The Mean Time Between Failures $(MTBF_i)$ for machine i, with a service time of t_i, is defined as $MTBF_i = t_i/f_i$, while the Mean Time To Repair $(MTTR_i)$ is defined as $MTTR_i = t_i/r_i$.

Referring to Figure 1, each buffer b_i, located immediately after the machine M_i, has a positive integer size. The total buffer capacity available for allocation across the buffers $b_1, b_2, \ldots, b_{n-1}$ in the serial production line is denoted as B_{max}. The objective of the BAP is to determine the vector $b = (b_1, b_2, \ldots, b_{n-1})$ that maximizes the average production rate while satisfying the constraint $\sum_{i=1}^{n-1} b_i = B_{max}$. The mathematical formulation of the BAP can be represented as follows:

$$\text{Find} \quad b = (b_1, b_2, \ldots, b_{n-1}) \ \textit{so as to maximise} \ f(b) \tag{1}$$

$$\text{Subject to:} \quad \sum_{i=1}^{n-1} b_i = B_{max};\ b_i \geq 0 \ \textit{and integer} \tag{2}$$

where $f(b)$ represents the average PR of the production line. Let us assume that the system operates for a duration of time T during an experiment, and during this time, the system produces a total of L parts. In this context, we can express the PR of the production line as follows:

$$PR = f(b) = L/T \tag{3}$$

IV. SOLUTION METHOD

A. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is a metaheuristic technique introduced by Kennedy and Eberhart [45]. It draws inspiration from the collective behavior of birds or fish flocks, where each particle represents a potential solution and learns from its own experience (personal best position) and the experiences of other particles (global best position) in the search space. The objective is to find the (near-)optimal solution through iterative movements. Various optimization algorithms may outperform PSO for specific scenarios. However, choosing PSO over other algorithms can be justified based on several factors. First, PSO stands out due to its simplicity and ease of implementation. It has a straightforward concept and requires minimal parameter tuning, making it accessible to researchers and practitioners. This simplicity facilitates its integration into various applications without extensive computational overhead or intricate implementation procedures. Second, PSO exhibits a desirable convergence speed, often enabling it to reach promising solution-space regions rapidly. This characteristic is advantageous when computing time is critical and prompt optimization results are required. Moreover, efforts to enhance PSO's performance by developing strategies that mitigate premature convergence issues will ensure a more effective search space exploration. Furthermore, PSO inherently balances exploration and exploitation by utilizing personal best and global best information. By leveraging these components with the introduction of novel concepts, PSO efficiently explores the search space while simultaneously exploiting promising regions. This ability to avoid getting trapped in local optima contributes to its effectiveness as an optimization algorithm.

In PSO, each particle is characterized by velocity and position. In the standard PSO algorithm, during each iteration or generation (denoted as *it*), the position (x_i^{it}) and velocity (v_i^{it}) of each particle *i* are updated according to Equations (4) and (5) below:

$$v_i^{it+1} = w\, v_i^{it} + c_1 R_1\left(p_{best_i}{}^{it} - x_i^{it}\right) + c_2\, R_2\left(p_{gbest}{}^{it} - x_i^{it}\right) \quad (4)$$

$$x_i^{it+1} = v_i^{it+1} + x_i^{it} \quad (5)$$

Here, *w* represents the inertia weight coefficient, while c_1 and c_2 are acceleration coefficients. R_1 and R_2 denote two random values sampled from the interval [0, 1]. Similarly, in subsequent equations (mentioned in the following subsection), $R_3, R_4, \ldots, R_{10}$ also represent random values from the interval [0, 1].

Equation (4) updates the velocity of the particle by considering three terms: the inertia term ($w\, v_i^{it}$), the cognitive term ($c_1 R_1\left(p_{best_i}{}^{it} - x_i^{it}\right)$), and the social term ($c_2\, R_2\left(p_{gbest}{}^{it} - x_i^{it}\right)$). The inertia term allows the particle to retain part of its previous velocity. The cognitive term adjusts the velocity based on the particle's personal best position, while the social term adjusts the velocity based on the global best position (p_{gbest}) among all particles. Equation (5) then updates the particle's position by adding the newly computed velocity. These iterative velocity and position updates allow the particles to explore and exploit the search space to converge toward the (near-)optimal solution.

B. PARAMETERS AND GENERATION OF THE INITIAL SWARM OF APSO

The following are the parameters of the proposed APSO method. Further information about these parameters can be found in Kassoul et al. [5].

- s_{wt}: social acceleration worst-coefficient
- c_{wt}: cognitive acceleration worst-coefficient
- c_b: cognitive acceleration best-coefficient
- s_b: social acceleration best-coefficient
- l_w: logarithmic weight parameter
- e_w: exponential weight parameter
- w: inertia weight
- N: population size
- d: damping coefficient
- c_1: cognitive scaling parameter
- c_2: social scaling parameter
- $maxG$: maximum number of generations (iterations)

The initial swarm consists of *m* different particles, denoted as $P = (P_1, P_2, \ldots, P_m)$. Each particle $P_i = \left(p_{i,1}, p_{i,2}, \ldots p_{i,j}, \ldots, p_{i,n-1}\right)$ represents a configuration with *n*-1 buffer capacities, where $p_{i,j}$ represents the j^{th} buffer (position) of the i^{th} particle. The particle's configurations are generated randomly and uniformly using the discrete uniform distribution, allowing for coverage of various regions in the search space. To satisfy the B_{max} constraint, an adjustment procedure is employed that uniformly increases or decreases specific buffer values of the particle. For a more comprehensive understanding of how these initial solutions are generated, please refer to Kassoul et al. [46].

C. PROPOSED APSO

This subsection provides a detailed description of the proposed APSO Algorithm, which serves two primary purposes. The first purpose is to overcome the limitations of classical PSO, such as premature convergence, by incorporating a jumping strategy consisting of a logarithmic part and an exponential part. The second purpose is to achieve faster convergence through velocity control using dynamic parameters. The pseudocode of APSO is presented in Algorithm 1.

The initial step of APSO involves generating the initial population swarm (as explained in Subsection IV.B) and assigning random velocities to the particles using the discrete uniform distribution $U\{v_{min}, v_{max}\}$ for each coordinate. Here, v_{min} and v_{max} represent the maximum and minimum values

of the velocity and are set to the lower and upper bounds of the variable search space, named l_b and u_b, respectively.

The next step of the algorithm involves decomposing the swarm population into three equal subswarms, namely N_1, N_2 and N_3, to effectively explore a large portion of the search space. The velocity of the first subswarm N_1 is formulated in Equation (8), incorporating exponential and logarithmic components denoted as ξ and ζ, respectively, and defined as:

$$\xi^{it} = e^{1/\left(\left\|p_i^{it} - x_i^{it}\right\| + \epsilon\right)} \tag{6}$$

$$\zeta^{it} = log\left(\left\|{p_{best}}_i^{it} - x_i^{it}\right\| + \epsilon\right) \tag{7}$$

where ϵ is a very small positive value.

The equation also includes various acceleration coefficients and random values ($R_{\text{vec}}, R_1, R_2, R_3, R_4$, and R_5) to influence the velocity update process.

$$\begin{aligned} v_i^{it+1} = {} & w\, v_i^{it} + e_w \xi R_{vec} + l_w \zeta R_1\left({p_{best}}_i^{it} - x_i^{it}\right) + \\ & c_b R_2\left({p_{best}}_i^{it} - x_i^{it}\right) + s_b R_3\left({p_{gbest}}^{it} - x_i^{it}\right) + \\ & c_{wt} R_4\left({p_{worst}}_i^{it} - x_i^{it}\right) + s_{wt} R_5\left({p_{gworst}}^{it} - x_i^{it}\right) \end{aligned} \tag{8}$$

The logarithmic and exponential components play a crucial role in allowing particles to jump when their velocity is near zero, indicating a possible entrapment in a local optimum. By escaping such regions, particles can explore the search space more effectively. To ensure the feasibility of the solution, the velocity and position equations of the proposed method will be updated as follows:

$$v_i^{it+1} = \begin{cases} v_{max} & if \;\; v_i^{it+1} > v_{max} \\ v_{min} & if \;\; v_i^{it+1} < v_{min} \\ v_i^{it+1} & otherwise \end{cases} \tag{9}$$

$$x_i^{it+1} = \begin{cases} u_b & if \;\; x_i^{it+1} > u_b \\ l_b & if \;\; x_i^{it+1} < l_b \\ x_i^{it+1} & otherwise \end{cases} \tag{10}$$

To strike a balance between exploration and exploitation, the exponential weight parameter e_w is utilized. It decreases linearly using the damping coefficient d, facilitating global exploration with large leaps at the beginning of the search and localized exploitation with small jumps in specific regions. The exponential weight parameter e_w is updated according to Equation (11).

$$e_w = d\, e_w \tag{11}$$

The velocity update of the first subswarm considers the global and personal worst positions to maintain population diversity and explore new regions in the solution space. Negative values are assigned to the coefficient parameters (c_{wt} and s_{wt}) associated with the personal and global worst positions, respectively, allowing for investigation of opposite solution-space regions visited by these positions.

For the second subpopulation N_2, the velocity update incorporates the logarithmic and exponential components, previous velocity, and global and personal best positions. The formulation of the velocity is given in Equation (12).

$$\begin{aligned} v_i^{it+1} = {} & w\, v_i^{it} + e_w \xi R_{vec} + l_w \zeta R_6\left({p_{best}}_i^{it} - x_i^{it}\right) + \\ & c_b R_7\left({p_{best}}_i^{it} - x_i^{it}\right) + s_b\, R_8\left({p_{gbest}}^{it} - x_i^{it}\right) \end{aligned} \tag{12}$$

Lastly, the velocity update of the subswarm N_3 follows the C_NPSO procedure proposed by [5] and differs from the standard PSO by setting $c_1 = -1$ and $c_2 = 2$. The equation for velocity update is provided in Equation (13).

$$\begin{aligned} v_i^{it+1} = {} & w v_i^{it} + c_1 R_9\left({p_{best}}_i^{it} - x_i^{it}\right) \\ & + c_2 R_{10}\left({p_{gbest}}^{it} - x_i^{it}\right) \end{aligned} \tag{13}$$

The value of the inertia weight parameter (w) in APSO decreases linearly from an initial value to a final value. This adjustment gradually reduces the particles' ability to perform a global search while increasing their ability to conduct a local search. The update of w is governed by Equation (14).

$$w = d\left(\frac{1-w}{1+w}\right) \tag{14}$$

Algorithm 1: APSO

Step1: **Initialization**
Set the initial values for the parameters $e_w, l_w, c_b, s_b, c_{wt}, s_{wt}, c_1, c_2, d, w, N, MaxG$.
Initialize the velocity v_i $(i = 1,2,\dots,N)$ uniformly and randomly.
Initialize the position x_i $(i = 1,2,\dots,N)$ uniformly and randomly.
Evaluate the fitness value of each particle (PR) using the simulation model, and set the initial best position values ${p_{best}}_i$ and worst position values ${p_{worst}}_i$ equal to the current x_i $(i = 1,2,\dots,N)$.
Determine the global worst and best positions p_{gworst} and p_{gbest}.

Step 2: **Update of the velocity and position of particles**
Divide the population into three equal subswarms N_1, N_2 and N_3.
for the first subswarm N_1, **do**
Calculate ξ and ζ using Equations (6) and (7).
Calculate the velocity v_i for each particle using Equation (8).

for the second subswarm N_2, **do**
Calculate the velocity v_i for each particle using Equation (12).

for the third subswarm N_3, **do**
Calculate the velocity v_i for each particle using Equation (13).
Apply the velocity limits using $v_{i,j} = \max(v_{i,j}; v_{min})$ and $v_i = \min(v_{i,j}; v_{max})$, where $v_{i,j}$ represents the j^{th} velocity coordinate of the i^{th} particle.
Update the position of particles using Equation (5).
Apply the position limits using $x_{i,j} = \max(x_{i,j}; l_b)$ and $x_i = \min(x_{i,j}; u_b)$, where $x_{i,j}$ represents the j^{th} position coordinate of the i^{th} particle.

Step 3: **Update of the personal and global worst/ best positions**
Calculate the fitness values (PR) for each particle.
Update the personal and global worst/ best positions of the particles.
Update the dynamic parameter e_w using Equation (11)
Update the inertia weight w using Equation (14)

Step 4: **Convergence procedure**
Check if the termination criterion is satisfied (i.e., verifying if the maximum number of iterations has been reached).

In summary, the APSO Algorithm encompasses a jumping strategy and dynamic parameter control to address the limitations of classical PSO. It divides the swarm into subswarms and employs different velocity update formulations for each. Incorporating logarithmic and exponential components, various acceleration coefficients, and dynamic parameter adjustments allow for effective exploration and exploitation of the search space. The proposed pseudocode in Algorithm 1 outlines the step-by-step implementation of APSO.

V. NUMERICAL EXPERIMENTS

The experiments are conducted on a computer with a 2.4-GHz Core (TM) i5 CPU and 16 GB of RAM. The discrete-event simulation models are developed using the Arena simulation language V14.0, and the algorithms are implemented in Java. The APSO algorithm is terminated after reaching a maximum of 50 generations ($MaxG = 50$).

The study considers three different sizes of serial production lines: small lines with 3, 5, or 10 machines, medium lines with 20 machines, and large lines with 40 or 100 machines. All machines in the production lines are identical and unreliable. The repair and failure times are geometrically distributed, with the MTTR calculated as $MTTR = 1/r$ and the MTBF calculated as $MTBF = 1/f$, where r and f are the repair and failure probabilities, respectively. The service time for each machine (t_i) is set to 1. The buffer allocation, average PR values, and convergence tendencies are analyzed in the study.

While the BAP in serial production lines has been extensively studied in the literature, few PSO approaches have been proposed for this problem. Therefore, APSO is compared with seven state-of-the-art algorithms that have demonstrated efficiency in solving the BAP, and for which data and results are available for some of the considered instances. The algorithms being compared are GT (Gradient Technique) [47], DGT (Dual Gradient Technique) [30], DC (Degraded Ceiling method and decomposition approximation) [32], IDA (Immune Decomposition Algorithm) [48], ADA-TS (Tabu Search with Analytical Decomposition Approximation) ([49], GA-SA (Simulated Annealing and Genetic Algorithm) [41], and GA-FPA (Genetic Algorithm and Finite Perturbation Analysis) [46]. GA-FPA is a highly efficient algorithm for the BAP involving serial production lines, and a detailed comparison with other methods is presented in Kassoul et al. [46].

TABLE I
NUMBER OF SIMULATED UNITS AND REPLICATIONS FOR THE SIX COMPARATIVE EXAMPLES

Instance size	3	5	10	20	40	100
Units	10,000	100,000	10,000	10,000	10,000	100,000
Replications	30	50	30	5	5	10

The best algorithm's results are highlighted in bold in the following tables (Tables III, V, VII to X). Figure 2 illustrates the convergence curves of APSO for representative instances with n values of 5, 10, 20, and 40. We highlight that all simulation runs in this study are performed using identical experimental conditions. Table I presents detailed information regarding the number of simulated units or parts, as well as the number of runs (replications) for each example. These specific numbers are carefully selected to ensure equitable comparisons with other methods and to obtain stable results with consistent average PR values.

Although a detailed discussion of computation times is not provided, it is worth noting that the average computation time to obtain the best solution is approximately 96 minutes for a large instance with n = 40 machines. This time frame is considered reasonable based on findings from several industrial studies in the production field [50]–[52]. It should be emphasized that this duration provides an order of magnitude estimate, indicating the time required for achieving optimal results in many real-world industrial scenarios.

A. RESULTS ON A SMALL INSTANCE WITH 3 MACHINES

Table II presents the repair and failure rates for the instance proposed by Gershwin and Schor [30], where the buffer configuration and production rate (PR) for this instance are not explicitly mentioned. Table III provides a comparison between IDA, GA-FPA, and APSO. Interestingly, APSO and GA-FPA yield the same PR, indicating their comparable performance in finding an optimal solution. Moreover, all three methods exhibit a similar buffer allocation scheme, where the first buffer is larger than the second buffer. This observation can be attributed to the relatively higher values of failure rate (f_1) and repair rate (r_1) associated with the first machine compared to the second machine (f_2 and r_2, respectively).

The preference for a larger buffer at the beginning of the production line can be explained by the need to accommodate potential disruptions caused by the first machine's higher failure and repair rates. By allocating a larger buffer, the system can mitigate the impact of downtime and maintain a smoother production flow.

TABLE II
PARAMETERS OF THE CONSIDERED INSTANCE

Machine	1	2	3
r_i	0.35	0.15	0.4
f_i	0.037	0.015	0.02

TABLE III
RESULTS ON AN INSTANCE WITH N = 3 MACHINES AND $B_{max} = 20$

Method	Buffer allocation		PR
IDA	14	6	0.86799
GA-FPA	13	7	**0.87178**
APSO	13	7	**0.87178**

B. RESULTS ON A SMALL INSTANCE WITH 5 MACHINES

Table IV provides the repair and failure rates for the instance proposed by Ho et al. [47]. Table V presents the results of GT, DGT, IDA, ADA-TS, GA-SA, GA-FPA and APSO. Among these methods, APSO achieves the highest PR value, indicating its superior performance in maximizing the PR. APSO reaches this optimal configuration within the first 15 generations, as illustrated in Figure 2.

An interesting observation from the results is that all the approaches, including APSO and the other algorithms, converge to configurations where smaller buffer slots are allocated at the ends of the production line. This allocation strategy is advantageous as it facilitates the smooth passage of parts and reduces the likelihood of line blockages. The system can maintain an uninterrupted production flow by allowing for smaller buffer slots at the ends, thereby improving overall productivity.

C. RESULTS ON A SMALL INSTANCE WITH 10 MACHINES

Table VI provides the repair and failure rates related explicitly to the instance proposed by Nahas et al. [32]. Table VII presents the comparative results of DC, IDA, ADA-TS, GA-SA, GA-FPA and APSO. By examining the data provided in Table VII, we can observe that APSO achieves the second-best PR value among the algorithms evaluated. Figure 2 highlights that APSO reaches its optimal buffer allocation solution relatively quickly. Specifically, the best buffer allocation for APSO is achieved before completing 25 generations, demonstrating its efficiency in finding the optimal solution quickly.

TABLE IV
PARAMETERS OF THE CONSIDERED INSTANCE

Machine	1	2	3	4	5
MTTR = $1/r_i$	11	19	12	7	7
MTBF = $1/f_i$	20	167	22	22	26

TABLE V
RESULTS ON AN INSTANCE WITH N = 5 MACHINES AND $B_{max} = 31$

Method	Buffer allocation	PR
GT	5 11 8 7	0.4914
DGT	7 10 10 4	0.4943
IDA	6 10 11 4	0.4941
ADA-TS	7 10 10 4	0.4943
GA-SA	7 10 10 4	0.4943
GA-FPA	7 11 9 4	0.4948
APSO	5 11 12 3	**0.4965**

TABLE VI
PARAMETERS OF THE CONSIDERED INSTANCE

Machine	1	2	3	4	5	6	7	8	9	10
MTTR =$1/r_i$	7	7	5	10	9	14	5	8	10	10
MTBF =$1/f_i$	20	30	22	22	25	40	23	30	45	20

TABLE VII
RESULTS ON AN INSTANCE WITH N = 10 MACHINES AND $B_{max} = 270$

Method	Buffer allocation	PR
DC	14 19 30 54 45 27 23 24 34	0.64135
IDA	14 19 30 52 47 27 23 24 34	0.64139
ADA-TS	14 19 30 54 45 27 23 24 34	0.64135
GA-SA	7 16 48 61 24 41 20 34 19	0.63016
GA-FPA	19 23 24 45 43 34 22 29 31	**0.64920**
APSO	13 24 27 42 44 35 24 30 31	0.64850

D. RESULTS ON MEDIUM INSTANCE WITH 20 MACHINES

In Table VIII, the first two columns present the repair and failure rates specifically for the instance proposed by Demir et al. [49]. In this instance, all machines exhibit similar repair and failure rates, ranging from 0.1 to 0.9, with an increment of 0.1. This implies that 9 cases are considered in this example, each corresponding to a different combination of repair and failure rates.

The subsequent columns in Table VIII provide the PR values obtained by the different methods: ADA-TS, GA-SA, GA-FPA, and APSO. Upon examination, it becomes evident that APSO outperforms both GA-SA and ADA-TS regarding PR values. This indicates that APSO consistently achieves higher-quality solutions compared to the other two methods for this instance. Interestingly, APSO and GA-FPA exhibit similar performances, as both methods reach the best results in four of the nine cases considered. This demonstrates their comparative effectiveness in finding optimal solutions.

Additionally, it is worth noting that APSO attains its best PR values before reaching generation 30, implying that it converges relatively quickly to high-quality solutions. This observation aligns with expectations, as it is generally anticipated that the PR values would increase with the values of r_i and f_i for all the methods tested.

TABLE VIII
RESULTS ON AN INSTANCE WITH N = 20 MACHINES AND $B_{max} = 100$

Parameters		**Average PR**			
r_i	f_i	APSO	GA-FPA	GA-SA	ADA-TS
0.1	0.1	0.221544	0.223559	0.226499	**0.234191**
0.2	0.2	0.304041	**0.305203**	0.302448	0.297025
0.3	0.3	**0.355727**	0.354811	0.349517	0.334450
0.4	0.4	0.393448	**0.394316**	0.354299	0.359637
0.5	0.5	0.421256	**0.422175**	0.382401	0.377895
0.6	0.6	**0.443987**	0.443916	0.397120	0.391846
0.7	0.7	**0.461614**	0.461407	0.446904	0.402760
0.8	0.8	0.476434	**0.476649**	0.450032	0.411638
0.9	0.9	**0.487833**	0.487760	0.459111	0.419018

TABLE IX
RESULTS ON AN INSTANCE WITH N = 40 MACHINES AND $B_{max} = 200$

Parameters		Average PR			
r_i	f_i	APSO	GA-FPA	GA-SA	ADA-TS
0.1	0.1	0.212987	0.214819	0.220132	**0.221273**
0.2	0.2	**0.293629**	0.293308	0.270289	0.282169
0.3	0.3	**0.345723**	0.344024	0.328518	0.325256
0.4	0.4	**0.383742**	0.383456	0.365301	0.351494
0.5	0.5	**0.412375**	0.412104	0.371244	0.370666
0.6	0.6	**0.435811**	0.435792	0.404937	0.385328
0.7	0.7	**0.455184**	0.454676	0.446904	0.397255
0.8	0.8	**0.470614**	0.470529	0.468270	0.406559
0.9	0.9	0.484953	**0.485077**	0.482113	0.413990

E. Results on large instances with 40 machines

Table IX follows the same structure as Table VIII and pertains to a larger instance proposed by Demir et al. [49]. The comparison involves the same methods as in Table VIII. However, it is noteworthy that APSO is notably more efficient than the other methods when confronted with this larger instance. In fact, APSO delivers the best solution in 7 out of the 9 cases evaluated. This finding suggests that APSO's efficiency improves as the instance size increases, making it a more effective approach than the alternative methods. Similar to the previous instance, it is worth noting that the PR exhibits an increasing trend concerning the value of r_i and f_i.

F. Results on a large instance with 100 machines

The study of BAPs for very large production lines has been relatively limited, as indicated by only a few dedicated studies, such as those conducted by Kose and Kilincci [41] and Spinellis et al. [18]. Researchers in this field have recognized that obtaining competitive results for such problems requires extensive computation times, often spanning several hours. The computation time exhibits exponential growth as the number of machines *n* increases. In this example, we focus on a production line of 100 unreliable machines proposed by Kose and Kilincci [41]. The repair and failure rates for these machines follow a geometric distribution with MTTR of 2 and MTBF of 20, respectively.

Due to the demanding nature of the required computations, the experiment is conducted with fewer generations. Expressly, MaxG is set to 20 instead of 50. Table X provides a comparative analysis of the results obtained from GA-SA and APSO algorithms. The computation times for each method are listed in seconds within the last column. Upon examination, it becomes evident that APSO outperforms GA-SA significantly regarding both solution quality (as indicated by PR values) and speed (as noted in the computing time required to obtain the best solution). This significant performance gap suggests that APSO offers a favorable trade-off between solution quality and computation time.

TABLE X
RESULTS ON AN INSTANCE WITH N = 100 MACHINES AND $B_{max} = 300$

Method	PR	CPU (s)
GA-SA	0.742554	125588.6
APSO	**0.759528**	**96750.1**

G. Behavior of the results when n augments

After analyzing Figure 2 and Tables III, V, VII to X, several observations can be made regarding the augmentation of n which represents the number of machines in the system. One notable finding is that APSO demonstrates increased competitiveness compared to other methods as n grows larger. APSO tends to achieve its best solutions later in optimization while maintaining a noticeable gap within the allowed number of generations (MaxG=50). For instance, in Figure 2, APSO reaches its optimal solution at generation 11 with n equal to 5, at generation 20 with n equal to 10, at generation 25 with n equal to 20, and at generation 22 with n equal to 40. This pattern suggests a high level of stability across different problem sizes, indicating that APSO effectively balances its exploration and exploitation search abilities throughout the search process.

Regarding allocating buffer capacity for production lines of varying sizes, APSO consistently generates the best buffer allocation in 14 out of the 22 cases, showcasing its superior performance. Even in cases where APSO is not ranked first, it still produces highly competitive results. However, the observed patterns of buffer allocation are often unexpected, underscoring the complexity of the problem. Notably, the buffer values in the middle tend to be similar and relatively larger compared to the buffer capacities at the ends of the production line. Additionally, as the number of machines increases, the buffer values tend to converge and become nearly identical.

In summary, the analysis of Figure 2 and Tables III, V, VII to X reveals that APSO exhibits enhanced competitiveness as the number of machines increases. It demonstrates stability across different problem sizes, effectively balancing exploitation and exploration abilities. APSO consistently generates optimal or highly competitive buffer allocations, highlighting its superior performance. The observed patterns in buffer allocation underscore the challenging nature of the problem, with middle buffers having similar and larger capacities compared to the end buffers and convergence of buffer values as the number of machines increases.

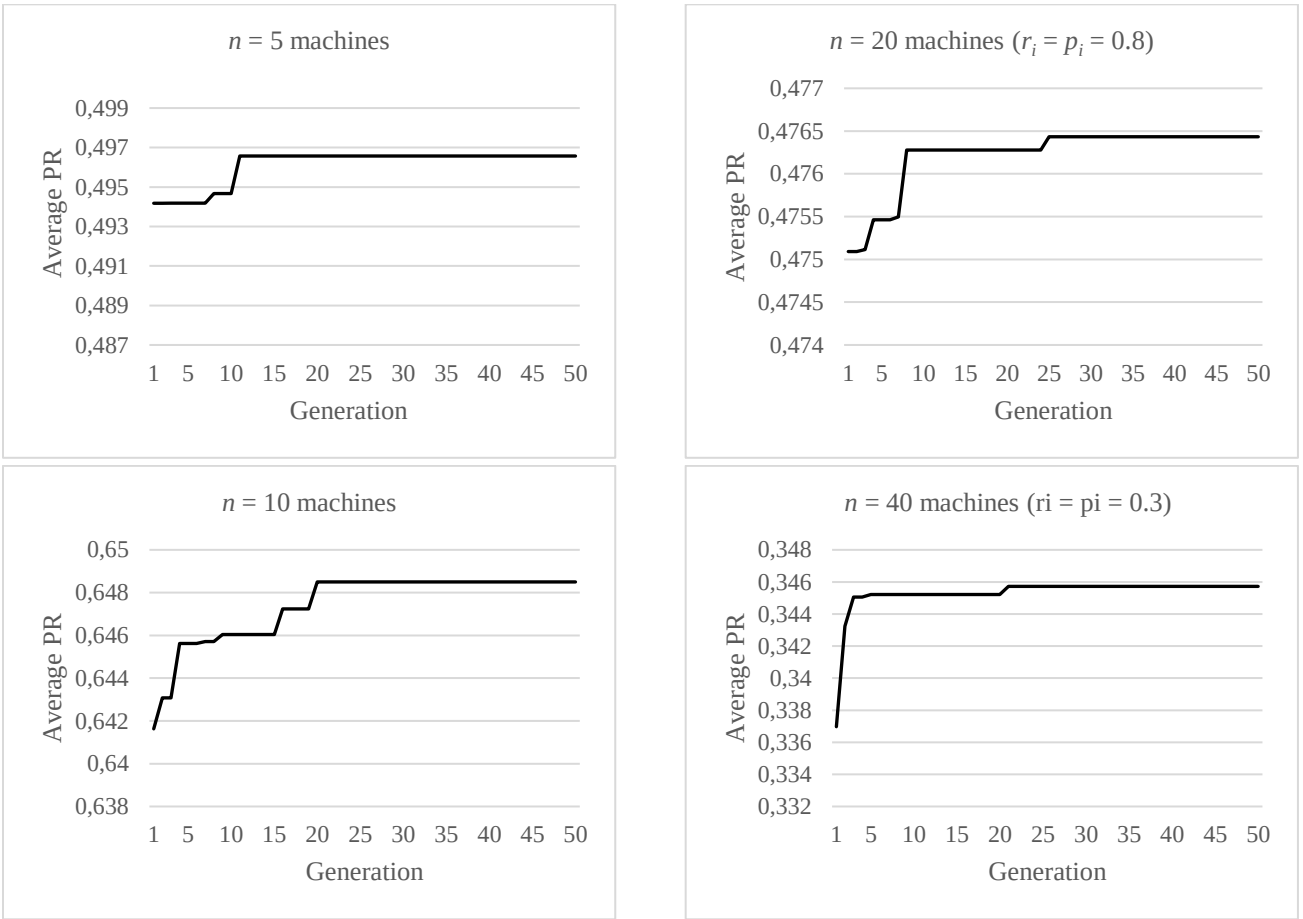


FIGURE 2. Convergence curves of APSO for various instances with n belonging to the set {5, 10, 20, 40}.

VI. CONCLUSION

This study introduces APSO, an efficient and reliable algorithm designed to address the buffer allocation problem in unreliable serial production lines to maximize the production rate. APSO is a variant of particle swarm optimization that incorporates a jumping strategy, utilizing exponential and logarithmic functions and dynamic parameters within the velocity equation. This strategy enables APSO to explore the solution space effectively and converge rapidly toward competitive solutions.

The experimental evaluation encompasses various instance sizes, ranging from small-scale systems with $n = 3$ machines to larger systems with $n = 100$ machines. The results demonstrate that APSO performs consistently well across different problem sizes, showcasing its ability to scale without diminishing performance. Notably, APSO outperforms state-of-the-art methods regarding solution quality and computation time, highlighting its superiority in solving the buffer allocation problem.

As a direction for future research, hybridizing APSO with other metaheuristics could be explored. This integration could involve incorporating refined learning mechanisms proposed by Thevenin and Zufferey [53] and by Schindl and Zufferey [54] within local-search approaches. Additionally, extending the proposed algorithm to more complex production lines, such as assembly and split systems, would be advantageous, further broadening the applicability and effectiveness of APSO in solving practical production optimization problems.

DISCLOSURE STATEMENT

The authors report there are no competing interests to declare.

FUNDING

This work is funded by the Swiss National Science Foundation under the grant 100018_182244.

DATA AVAILABILITY STATEMENT

The paper includes no data.

ACKNOWLEDGEMENT

This paper is based on chapter 2 of the PhD Thesis of the first author [55].

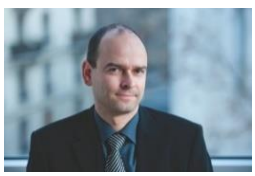
REFERENCES

[1] J. Li and S. M. Meerkov, "Mathematical Modeling of Production Systems," in *Production Systems Engineering*, J. Li and S. M. Meerkov, Eds., Boston, MA: Springer US, 2009, pp. 1–59. doi: 10.1007/978-0-387-75579-3_3.
[2] L. Demir, S. Tunali, and D. T. Eliiyi, "The state of the art on buffer allocation problem: a comprehensive survey," *J. Intell. Manuf.*, vol. 25, no. 3, pp. 371–392, 2014.
[3] S. Weiss, J. A. Schwarz, and R. Stolletz, "The buffer allocation problem in production lines: Formulations, solution methods, and instances," *IISE Trans.*, vol. 51, no. 5, pp. 456–485, 2019.
[4] J. MacGregor Smith and F. R. Cruz, "The buffer allocation problem for general finite buffer queueing networks," *IIE Trans.*, vol. 37, no. 4, pp. 343–365, 2005.
[5] K. Kassoul, N. Zufferey, N. Cheikhrouhou, and S. Brahim Belhaouari, "Exponential Particle Swarm Optimization for Global Optimization," *IEEE Access*, vol. 10, pp. 78320–78344, 2022, doi: 10.1109/ACCESS.2022.3193396.
[6] K. Kassoul, S. B. Belhaouari, and N. Cheikhrouhou, "Dynamic Cognitive-Social Particle Swarm Optimization," in *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, Feb. 2021, pp. 200–205. doi: 10.1109/ICARA51699.2021.9376550.
[7] K. L. Narasimhamu, V. V. Reddy, and C. S. P. Rao, "Optimal Buffer Allocation in Tandem Closed Queuing Network with Single Server Using PSO," *Procedia Mater. Sci.*, vol. 5, pp. 2084–2089, Jan. 2014, doi: 10.1016/j.mspro.2014.07.543.
[8] K. L. Narasimhamu, V. V. Reddy, and C. S. P. Rao, "Optimization of Buffer Allocation in Manufacturing System Using Particle Swarm Optimization," *Int. Rev. Model. Simul. IREMOS*, vol. 8, no. 2, Art. no. 2, Apr. 2015, doi: 10.15866/iremos.v8i2.5666.
[9] A. K. Tsadiras, C. T. Papadopoulos, and M. E. O'Kelly, "An artificial neural network based decision support system for solving the buffer allocation problem in reliable production lines," *Comput. Ind. Eng.*, vol. 66, no. 4, pp. 1150–1162, 2013.
[10] A. Alfieri and A. Matta, "Mathematical programming formulations for approximate simulation of multistage production systems," *Eur. J. Oper. Res.*, vol. 219, no. 3, pp. 773–783, 2012.
[11] G. Pedrielli, A. Alfieri, and A. Matta, "Integrated simulation–optimisation of pull control systems," *Int. J. Prod. Res.*, vol. 53, no. 14, pp. 4317–4336, 2015.
[12] D. D. Spinellis and C. T. Papadopoulos, "A simulated annealing approach for buffer allocation in reliable production lines," *Ann. Oper. Res.*, vol. 93, no. 1, pp. 373–384, 2000.
[13] A. C. Diamantidis and C. T. Papadopoulos, "Exact analysis of a two-workstation one-buffer flow line with parallel unreliable machines," *Eur. J. Oper. Res.*, vol. 197, no. 2, pp. 572–580, 2009.
[14] A. Dolgui, A. Eremeev, and V. Sigaev, "On Local Optima Distribution in Buffer Allocation Problem for Production Line with Unreliable Machines," *IFAC-Pap.*, vol. 55, no. 10, pp. 1092–1097, 2022.
[15] M. Amiri and A. Mohtashami, "Buffer allocation in unreliable production lines based on design of experiments, simulation, and genetic algorithm," *Int. J. Adv. Manuf. Technol.*, vol. 62, no. 1, pp. 371–383, 2012.
[16] S. Weiss, A. Matta, and R. Stolletz, "Optimisation of buffer allocations in flow lines with limited supply," *IISE Trans.*, vol. 50, no. 3, pp. 191–202, 2018.
[17] S. Helber, K. Schimmelpfeng, R. Stolletz, and S. Lagershausen, "Using linear programming to analyze and optimize stochastic flow lines," *Ann. Oper. Res.*, vol. 182, no. 1, pp. 193–211, 2011.
[18] D. Spinellis, C. Papadopoulos, and J. M. Smith, "Large production line optimization using simulated annealing," *Int. J. Prod. Res.*, vol. 38, no. 3, pp. 509–541, 2000.
[19] H.-Y. Zhang, Q.-X. Chen, J. M. Smith, N. Mao, A.-L. Yu, and Z.-T. Li, "Performance analysis of open general queuing networks with blocking and feedback," *Int. J. Prod. Res.*, vol. 55, no. 19, pp. 5760–5781, 2017.
[20] A. Diamantidis, J.-H. Lee, C. T. Papadopoulos, J. Li, and C. Heavey, "Performance evaluation of flow lines with non-identical and unreliable parallel machines and finite buffers," *Int. J. Prod. Res.*, vol. 58, no. 13, pp. 3881–3904, 2020.
[21] G. Liberopoulos, "Performance evaluation of a production line operated under an echelon buffer policy," *IISE Trans.*, vol. 50, no. 3, pp. 161–177, 2018.
[22] S. Xi, Q. Chen, J. MacGregor Smith, N. Mao, A. Yu, and H. Zhang, "A new method for solving buffer allocation problem in large unbalanced production lines," *Int. J. Prod. Res.*, vol. 58, no. 22, pp. 6846–6867, 2020.
[23] Y. Bai, J. Tu, M. Yang, L. Zhang, and P. Denno, "A new aggregation algorithm for performance metric calculation in serial production lines with exponential machines: design, accuracy and robustness," *Int. J. Prod. Res.*, vol. 59, no. 13, pp. 4072–4089, 2021.
[24] O. Abdelrahman and P. Keikhosrokiani, "Assembly Line Anomaly Detection and Root Cause Analysis Using Machine Learning," *IEEE Access*, vol. 8, pp. 189661–189672, 2020, doi: 10.1109/ACCESS.2020.3029826.
[25] C. T. Papadopoulos, M. E. J. O'Kelly, and A. K. Tsadiras, "A DSS for the buffer allocation of production lines based on a comparative evaluation of a set of search algorithms," *Int. J. Prod. Res.*, vol. 51, no. 14, pp. 4175–4199, 2013.
[26] F. S. Hillier and K. C. SO, "The effect of the coefficient of variation of operation times on the allocation of storage space in production line systems," *IIE Trans.*, vol. 23, no. 2, pp. 198–206, 1991.
[27] W. Zhou and Z. Lian, "A tandem network with a sharing buffer," *Appl. Math. Model.*, vol. 35, no. 9, pp. 4507–4515, 2011.
[28] A. C. Diamantidis and C. T. Papadopoulos, "A dynamic programming algorithm for the buffer allocation problem in homogeneous asymptotically reliable serial production lines," *Math. Probl. Eng.*, vol. 2004, no. 3, pp. 209–223, 2004.
[29] T. Alfakih, M. M. Hassan, and M. Al-Razgan, "Multi-Objective Accelerated Particle Swarm Optimization With Dynamic Programing Technique for Resource Allocation in Mobile Edge Computing," *IEEE Access*, vol. 9, pp. 167503–167520, 2021, doi: 10.1109/ACCESS.2021.3134941.
[30] S. B. Gershwin and J. E. Schor, "Efficient algorithms for buffer space allocation," *Ann. Oper. Res.*, vol. 93, no. 1, pp. 117–144, 2000.
[31] C. Shi and S. B. Gershwin, "A segmentation approach for solving buffer allocation problems in large production systems," *Int. J. Prod. Res.*, vol. 54, no. 20, pp. 6121–6141, 2016.
[32] N. Nahas, D. Ait-Kadi, and M. Nourelfath, "A new approach for buffer allocation in unreliable production lines," *Int. J. Prod. Econ.*, vol. 103, no. 2, pp. 873–881, 2006.
[33] M. Qaraad, S. Amjad, N. K. Hussein, S. Mirjalili, N. B. Halima, and M. A. Elhosseini, "Comparing SSALEO as a Scalable Large Scale Global Optimization Algorithm to High-Performance Algorithms for Real-World Constrained Optimization Benchmark," *IEEE Access*, vol. 10, pp. 95658–95700, 2022, doi: 10.1109/ACCESS.2022.3202894.
[34] I. Sabuncuoglu, E. Erel, and Y. Gocgun, "Analysis of serial production lines: characterisation study and a new heuristic procedure for optimal buffer allocation," *Int. J. Prod. Res.*, vol. 44, no. 13, pp. 2499–2523, 2006.
[35] L. Demir, A. C. Diamantidis, D. T. Eliiyi, M. E. O'Kelly, and S. Tunali, "Optimal buffer allocation for serial production lines using heuristic search algorithms: A comparative study," *Int. J. Ind. Syst. Eng.*, vol. 33, no. 2, pp. 252–270, 2019.
[36] Y. Alaouchiche, Y. Ouazene, and F. Yalaoui, "Multi-Objective Optimization of Energy-Efficient Buffer Allocation Problem for Non-Homogeneous Unreliable Production Lines," *IEEE Access*, vol. 10, pp. 3320–3335, 2022, doi: 10.1109/ACCESS.2021.3139954.
[37] X. Han, Y. Dong, L. Yue, and Q. Xu, "State Transition Simulated Annealing Algorithm for Discrete-Continuous Optimization Problems," *IEEE Access*, vol. 7, pp. 44391–44403, 2019, doi: 10.1109/ACCESS.2019.2908961.
[38] S. Gao, "A Bottleneck Detection-Based Tabu Search Algorithm for the Buffer Allocation Problem in Manufacturing Systems," *IEEE Access*, vol. 10, pp. 60507–60520, 2022, doi: 10.1109/ACCESS.2022.3181134.
[39] H. Peng, C. Ying, S. Tan, B. Hu, and Z. Sun, "An Improved Feature Selection Algorithm Based on Ant Colony Optimization," *IEEE Access*, vol. 6, pp. 69203–69209, 2018, doi: 10.1109/ACCESS.2018.2879583.
[40] L. Shi and S. Men, "Optimal buffer allocation in production lines," *IIE Trans.*, vol. 35, no. 1, pp. 1–10, 2003.
[41] S. Y. Kose and O. Kilincci, "Hybrid approach for buffer allocation in open serial production lines," *Comput. Oper. Res.*, vol. 60, pp. 67–78, 2015.
[42] J. T. Lin and C.-C. Chiu, "A hybrid particle swarm optimization with local search for stochastic resource allocation problem," *J. Intell. Manuf.*, vol. 29, no. 3, pp. 481–495, 2018.
[43] K. Kassoul, N. Cheikhrouhou, and N. Zufferey, "Simultaneous allocation of buffer capacities and service times in unreliable production

lines," *Int. J. Prod. Res.*, vol. 0, no. 0, pp. 1–21, Jan. 2023, doi: 10.1080/00207543.2023.2168310.
[44] N. Zufferey, "Optimization by ant algorithms: possible roles for an individual ant," *Optim. Lett.*, vol. 6, no. 5, pp. 963–973, 2012.
[45] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, IEEE, 1995, pp. 1942–1948.
[46] 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.
[47] Y.-C. Ho, M. A. Eyler, and T.-T. Chien, "A gradient technique for general buffer storage design in a production line," *Int. J. Prod. Res.*, vol. 17, no. 6, pp. 557–580, 1979.
[48] Y. Massim, F. Yalaoui, L. Amodeo, E. Chatelet, and A. Zeblah, "Efficient combined immune-decomposition algorithm for optimal buffer allocation in production lines for throughput and profit maximization," *Comput. Oper. Res.*, vol. 37, no. 4, pp. 611–620, 2010.
[49] L. Demir, S. Tunali, and A. Løkketangen, "A tabu search approach for buffer allocation in production lines with unreliable machines," *Eng. Optim.*, vol. 43, no. 2, pp. 213–231, 2011.
[50] J. Respen, N. Zufferey, and E. Amaldi, "Metaheuristics for a job scheduling problem with smoothing costs relevant for the car industry," *Networks*, vol. 67, no. 3, pp. 246–261, 2016.
[51] N. Zufferey, "Tabu Search Approaches for Two Car sequencing problems with smoothing constraints," in *Metaheuristics for Production Systems*, Springer, 2016, pp. 167–190.
[52] 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, pp. 1195–1210, 2017.
[53] S. Thevenin and N. Zufferey, "Learning variable neighborhood search for a scheduling problem with time windows and rejections," *Discrete Appl. Math.*, vol. 261, pp. 344–353, 2019.
[54] D. Schindl and N. Zufferey, "A learning tabu search for a truck allocation problem with linear and nonlinear cost components," *Nav. Res. Logist. NRL*, vol. 62, no. 1, pp. 32–45, 2015, doi: 10.1002/nav.21612.
[55] K. Kassoul, "Hybrid algorithms for designing production lines," Doctoral dissertation, University of Geneva, 2022.

Khelil Kassoul is a researcher at the Geneva School of Business Administration (HEG) – University of Applied Sciences of Western Switzerland (HES-SO). He obtained his PhD in Economics and Management from GSEM (Geneva School of Economics and Management). Previously, he worked as an associate member at CERN (European Organization for Nuclear Research). From 2003 to 2021, he held the position of teacher of mathematics and physics for Swiss maturity and Baccalaureate classes. His research focuses on various areas, including applied mathematics, production systems modeling, simulation and optimization, inventory management, and artificial intelligence.

Naoufel Cheikhrouhou is Professor of Supply chain, Logistics and Operations Management at Geneva School of Business Administration University of Applied Sciences Western Switzerland. Previously, he was leading the Operations Management research group at the Swiss Federal Institute of Technology at Lausanne (EPFL). He graduated from the School of Engineers of Tunis and received his PhD in Industrial Engineering from Grenoble Institute of Technology in 2001. His main research interests are modelling, simulation and optimisation of enterprise networks, behavioral operations management and Demand planning and forecasting. Dr. Cheikhrouhou 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, Naoufel Cheikhrouhou published more than 100 papers in scientific journals and conferences and regularly acts as keynote speaker in academic and industrial international conferences.



Nicolas Zufferey is a full professor of operations management at the *University of Geneva* in Switzerland, since 2008. His research activity focuses on designing solution methods for difficult and large optimization problems, with applications mainly in transportation, scheduling, production, inventory management, network design, and supply chain management. He is member of the *CIRRELT* transportation and logistics research center (www.cirrelt.ca) and of the *GERAD* decision-analysis research center (www.gerad.ca). He received his BSc and MSc degrees in Mathematics at EPFL (*Swiss Federal Institute of Technology* at Lausanne), as well as his PhD degree in operations research (2002). He was then successively a post-doctoral trainee at the *University of Calgary* (2003 – 2004) and an assistant professor at *Laval University* (2004 – 2007). He is the (co)author of more than 140 publications (papers in professional journals, proceedings of conferences, and book chapters). He has had research activities with 32 Universities in Europe and America, as well as with 26 private companies.