INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH
https://doi.org/10.1080/00207543.2021.1909169

Taylor & Francis
Taylor & Francis Group

Buffer allocation design for unreliable production lines using genetic algorithm and finite perturbation analysis

Khelil Kassoul[a,b], Naoufel Cheikhrouhou [b] and Nicolas Zufferey[a]

[a]Geneva School of Economics and Management, GSEM – University of Geneva, Geneva, Switzerland; [b]Geneva School of Business Administration, University of Applied Sciences Western Switzerland (HES-SO), Geneva, Switzerland

ABSTRACT

The buffer allocation problem in production lines is an NP-hard combinatorial optimisation problem. This paper proposes a new hybrid optimisation approach (using simulation) relying on genetic algorithm (GA) and finite perturbation analysis (FPA). Unlike the infinitesimal perturbation analysis, which deals with small (infinitesimal variation) perturbations for estimating gradients of the performance measure, FPA deals with larger (finite) or more lasting perturbations. It is an extension specifically dedicated to discrete decision variables and applicable to most discrete-event dynamic systems. The proposed method allows a global search using GA, with refinement in specific solution-space regions using FPA. The main objective is to maximise the average production rate of a production line with unreliable machines, by allocating the total buffer capacity in locations between machines. Extensive numerical experiments show that: (1) the proposed hybrid GA-FPA method clearly outperforms the state-of-the-art methods from the literature; (2) combining FPA and GA is beneficial when compared to employing GA or FPA independently.

ARTICLE HISTORY
Received 7 September 2020
Accepted 12 March 2021

KEYWORDS
Buffer allocation; production line design; genetic algorithm; perturbation analysis; simulation; stochastic optimisation

1. Introduction

The design of unreliable manufacturing systems, such as production lines with human operators or automated assembly systems, has received considerable attention from both academics and industrial worlds. Numerous researches are dedicated to the *Buffer Allocation Problem* (BAP), that is, how much buffer capacities to allow and where to place them within the line (Dolgui, Eremeev, and Sigaev 2007). A classification review of the studies on BAPs is presented in (Weiss, Schwarz, and Stolletz 2018) and in (Demir, Tunali, and Eliiyi 2014).

Most of the methods for designing production lines are based on flow simulations that require numerous iterations (Shi and Men 2003). In the industry, it is difficult to implement such methods since running them requires a huge amount of time. Therefore, this study focuses on reducing the time needed to reach convergence, while trying to reach the optimal buffer allocation scheme. An efficient technique that shows good convergence features is the *Finite Perturbation Analysis* (FPA) method. This method is an analytic technique that provides a solution to the design problem after a single simulation run of the system's model (Suri 1989). This paper proposes a hybrid method using the *Genetic Algorithm* (GA) and the FPA approaches, coupled to discrete-event simulation technique. The advantage of using GA lies in the fact that it offers a strong diversification ability to explore the solution space. GA is coupled here with FPA to allow an intensification of the search in some regions of the solution space. An exploration direction in the searched area is determined by calculating the expectation of the gradient of the performance measure (in this paper, the performance measure is the production rate) with respect to the system parameters. The gradient is used in an allocation technique of buffer capacities for unreliable production lines operating in a stochastic environment. The advantage of the technique is that it converges in a single simulation run and ensures scalability, as it can address different sizes of production lines (with different numbers of machines and buffers). To the best of our knowledge, combining GA with FPA has never been addressed in the literature, which shows the novelty of our work.

The paper is organised as follows. Section 2 gives the formulation of the problem and reviews the related work of BAP in production lines. In Section 3, we present the overall approach for solving the BAP. Section 4 gives a detailed development of the used optimisation

CONTACT Naoufel Cheikhrouhou naoufel.cheikhrouhou@hesge.ch

techniques. Section 5 presents the experiments, the results and their discussions. Finally, conclusions and future research directions are provided in Section 6.

2. Presentation of the problem and related work

In this section, we first explain the BAP in serial production lines and we formulate the problem. Second, we provide a comprehensive literature review.

2.1. Problem description and formulation

Consider a serial production line that consists of a sequence $(M_1, \ldots, M_n)$ of n machines connected in series. The machines are separated by $(n\text{-}1)$ buffers $(b_1, \ldots, b_{n-1})$, where $b_i$represents the size of the buffer located between the machines M_i and M_{i+1}, B_{max} is a fixed non-negative integer that represents the total buffer space available for the whole production line, and f (s) is a mathematical function that represents the *Production Rate* (*PR*) of the production line (see Figure 1). Each part enters the system through the first machine, passes sequentially through all machines and the intermediate buffers, and then exits the line through the last machine. In open production lines, the first machine is never starved (i.e. there are always available parts at the input of the system), and the last machine is never blocked (i.e. there is available space for parts storage at the output of the system). It is assumed that the transfer times to move parts from a machine to a buffer and vice versa are negligible.

Formulation: The design problem considered in this paper consists in allocating B_{max} throughout $(n\text{-}1)$ buffers. The objective is to maximise the average production rate in open serial production lines. In mathematical terms, the problem can be written as follows:

$$\text{Find: } s = (b_1, b_2, \ldots, b_{n-1}) \text{ so as to maximize } f(s) \tag{1}$$

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

2.2. Literature review

The design of production lines is a well-known problem in industry. It can be separated into two main categories. The first category focuses on designing the total cycle time across workstations, whereas the second one (namely, the BAP) focuses on buffer allocation between machines to store work in process. Regarding the latter, several authors address this problem for small-sized production lines by using analytical models and techniques. Markov chain models are one of the analytical techniques that have been widely addressed in the literature. To reach the optimal buffer allocation, Hillier and So (1991) use an exact analytic model based on Markov-chain representations for small-sized production lines. However, they show that any attempt to expand the model to a realistic situation leads to state space problems. Diamantidis and Papadopoulos (2009) study a serial flow line with two workstations and an intermediate buffer. They propose an analytical approach based on a Markov process model to describe the characteristics of the system. For more details on timed Markov models of manufacturing systems, the reader can refer to (Papadopoulos, Li, and O'Kelly 2019).

To evaluate large systems, approximation methods are developed relying on decomposition or aggregation methods. These methods are applied if a huge state space makes the numerical solution of Markov chains too slow or impossible (Weiss, Schwarz, and Stolletz 2018). The decomposition method (Liberopoulos 2018; Diamantidis et al. 2019; Shaohui et al. 2019) is based on a decomposition of the n-machine line into a set of $(n\text{-}1)$ two-machine lines where the buffer between the upstream machine and the downstream machine has the same capacity. On the other hand, the aggregation methods (Zhao et al. 2017) replace a two-machine line with an equivalent single machine, and then reduce iteratively the number of machines until obtaining only one machine. This reduction allows for a fast performance evaluation of long lines. Recently, some researchers use metaheuristics to solve BAPs for reliable or unreliable machines. For details about metaheuristics, the reader can find some principles and rules in (Zufferey 2012; Gendreau and Potvin 2019). Nahas, Ait-Kadi, and Nourelfath (2006) propose a new local search approach based on the degraded ceiling metaheuristic to solve the BAP in unreliable production

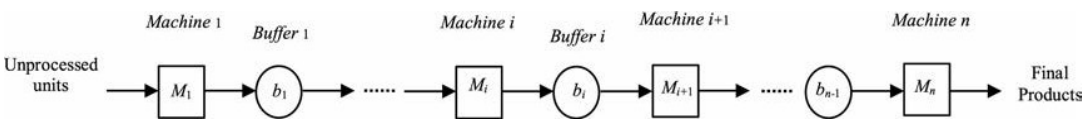

Figure 1. Open serial production line.

lines. They conclude that the degraded ceiling gives better results in an acceptable amount of time, and as the size of the line increases, the solutions become more competitive. Spinellis and Papadopoulos (2000) apply two stochastic methods, GA and simulated annealing, for solving the BAP with the objective of maximising the production rate. They show that both methods can be used for optimising large production lines. Spinellis, Papadopoulos, and MacGregor (2000) present a generalised queuing network algorithm as an evaluative procedure for optimising production-line configurations using simulated annealing. Dolgui et al. (2002) propose an approximate method based on the Markov-model aggregation approach using a GA. They obtain better solutions in comparison with other methods. Costa et al. (2015) propose a new parallel tabu-search algorithm using an adaptive neighbourhood generation mechanism, which consists in minimising the total buffer capacity of a serial production system under a minimum *PR* constraint. In another study, Vergara and Kim (2009) develop a buffer allocation method that can be applied to open or closed production lines. They use a heuristic to find a near-optimal solution and the performance of the heuristic is compared to the performance of GA for various test cases. Demir, Tunali, and Eliiyi (2011) use an adaptive tabu-search algorithm where the *PR* of the production line is evaluated by using a decomposition method. Recently, Cruz, Duarte, and Souza (2018) develop a multi-objective GA to allocate the buffers and service rates to generate a set of solutions for more than one objective functions. Yegul et al. (2017) focus on the problem of optimising production-line configurations and propose several simulation-based optimisation approaches based on ant-colony optimisation. Kose and Kilincci (2020) combine two metaheuristics and propose a hybrid-evolutionary simulation-optimisation approach for the multi-objective BAP in open serial production lines.

In other studies, authors propose to hybridise metaheuristics with other search techniques. Dolgui, Eremeev, and Sigaev (2007) present a hybrid optimisation algorithm, using GA and branch-and-bound approaches. For the evaluation of objective function, they propose a Markov-model aggregation technique. Bierlaire, Thémans, and Zufferey (2010) have combined variable neighbourhood search (Thevenin and Zufferey 2019) (with the aim to explore the solution space) with a modified trust-region algorithm (for intensification purposes), for nonlinear global optimisation. For solving the BAP in unreliable machines, Shi and Men (2003) propose a tabu search (Schindl and Zufferey 2015) to speed up the nested partitions method. Amiri and Mohtashami (2011) propose a GA integrated to line-search method for determining the optimal/near-optimal size of each buffer storage. According to Demir, Tunali, and Eliiyi (2014), only 4 out of 95 studies after 1998 employ a hybrid approach to optimise the buffer sizes, and just one of those researches solves the problem in a multi-objective manner whereas the others solve the BAP while maximising the *PR*. The reader can refer to Li et al. (2009) for more details on the *PR* evaluation. They review and summarise the most important publications and methods dealing with the *PR* analysis of production systems with unreliable machines and finite buffer capacities.

With respect to the objective function, the majority of studies done in the literature deal with a single objective problem. Some papers have addressed a bi-objective problem formulation (Lee, Chen, and Shunder 2009; Amiri and Mohtashami 2011; Cruz et al. 2012), and other studies involve three objectives (Gershwin and Schor 2000; Cruz, Van Woensel, and Smith 2010; Zandieh, Joreir-Ahmadi, and Fadaei-Rafsanjani 2017). This observation is also true for the job-scheduling field, where only a few studies have considered jointly more than two objective-function components (e.g. Thevenin, Zufferey, and Widmer 2016; Thevenin, Zufferey, and Potvin 2017).

Simulation-optimisation approaches are often employed for production lines. The simulation allows evaluating, at each execution, the stationary performance measure, whereas the optimisation algorithm provides a direction for searching new solutions for the design of the system. Battini, Persona, and Regattieri (2009) use a simulation approach to determine the optimal buffer allocation for a serial production line that increases the reliability of the production line by minimising the micro-downtime of machines. For their simulation, they consider the impact of the cost on the buffer allocation. BCan and Heavey (2011) employ a genetic programming to perform symbolic regression. They develop a sampling approach adapted to genetic programming, obtaining simulation-based meta-models of industrial systems. Gürkan (2000) optimises the *PR* with respect to buffer capacities using sample-path optimisation to find optimal buffer allocations by working with a continuous-line approach instead of discrete lines. This is because continuous-line simulations are substantially faster and the approximations are quite accurate. Dealing with continuous parameters enables him to compute directional derivatives of the *PR* that are valuable for optimisation purposes using infinitesimal perturbation analysis (Turki, Hennequin, and Sauer 2013). Cheikhrouhou, Paris, and Pierreval (2002) propose an algorithm designed to determine simultaneously the sizes of storage space and service times in

stochastic transfer lines. Based on FPA, the developed technique retrieves information on throughput gradient with respect to system parameters. The provided information is integrated into a multidimensional optimisation procedure to determine the best configuration of the system parameters.

Classical simulation-optimisation methods and techniques based on evolutionary algorithms (Fu 1994) consume significant computing time. Indeed, they require a large number of simulations before an optimal/near-optimal solution can be found. Therefore, this work uses FPA that provides a solution to the design problem using a single long simulation of the system, significantly reducing the computation time. The advantage of FPA is to be able to use the same simulation to estimate the gradients and to develop the configuration using a stochastic algorithm. Indeed, during the simulation, it is possible to obtain updated values of the gradient of the considered performance measure at regular time intervals. At each update, these values are used to guide the search for solutions throughout the simulation.

Although hybridising different methods has been proposed to solve the BAP, to the best of our knowledge, we could not find any study combining GA and FPA to solve the BAP in unreliable production lines for maximising the production rate. The proposed approach exploits the advantages of the two algorithms. GA tends to quickly identify promising regions of the solution space, whereas FPA deeply explores such regions to find competitive solutions. Moreover, an experimental study is carried out to identify the best parameters of the proposed method. The algorithms are applied equally to short and large production lines. A contribution of our work is thus the development of a general optimisation approach (GA combined with FPA) using a single simulation for the design of buffer capabilities for stochastic production lines. Another advantage of this approach is that it can take into account different sizes of production system.

3. Problem solving framework

The proposed approach is based on a combination of GA with FPA. Figure 2 represents the detailed optimisation approach, where the same FPA algorithm forms the core element providing the different gradients estimates for the *Stochastic Algorithm* (SA). The advantage of using GA is to approach competitive solutions in a reasonable time. The interest of FPA is to use, on the one hand, the same simulation to estimate the gradients, and on the other hand, to develop the configuration with a SA. In other words, the optimisation and the simulation take place simultaneously. In the initial population of m different solutions, $P = (s_1, s_2, \ldots, s_m)$, each solution $s_i = (b_{i,1}, b_{i,2}, \ldots b_{i,j}, \ldots, b_{i,n-1})$ is generated randomly from a discrete uniform distribution over nonnegative integers, where $b_{i,j}$ denotes the jth buffer of the i^{th} solution. The evolution of the current population P to the new population $P' = (s'_1, s'_2, \ldots, s'_m)$ is achieved by the application of the selection and crossover operators for a number

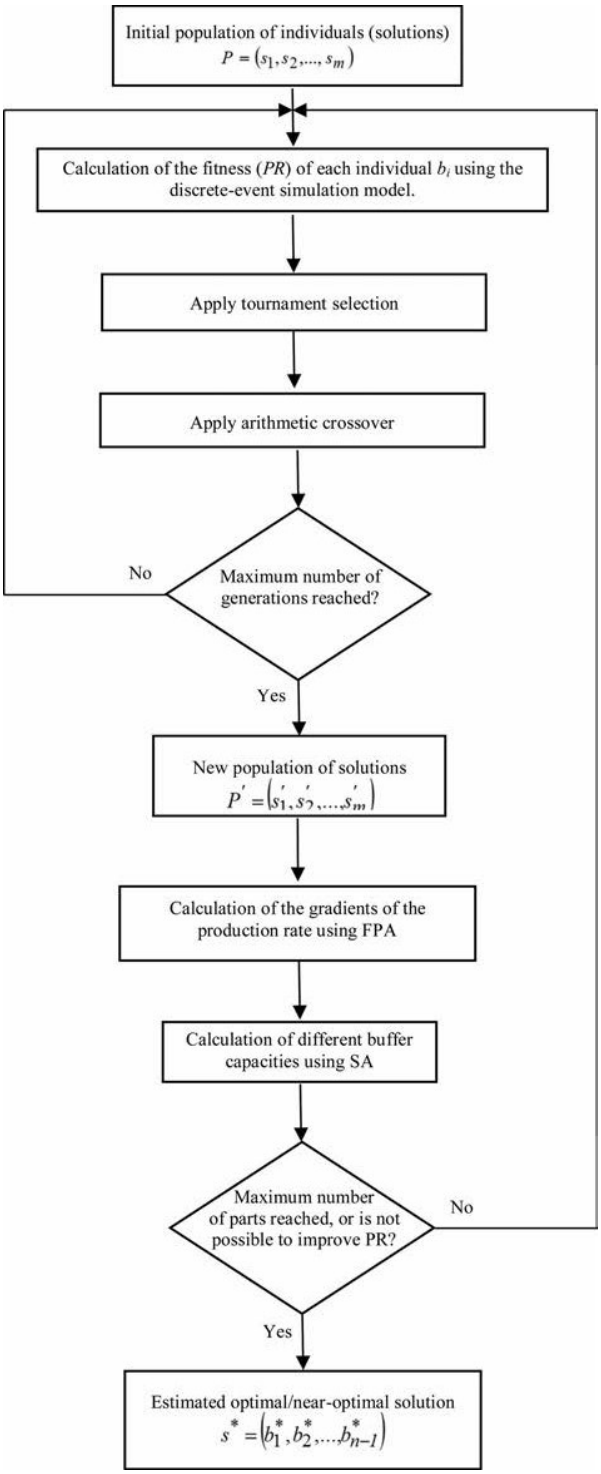

Figure 2. Framework of our solution approach for the BAP.

of generations, with the objective of approaching the zone containing an optimal/near-optimal solution $s^* = (b_1^*, b_2^*, \ldots, b_{n-1}^*)$. Each solution (or configuration) of P' is the input of the SA based on the calculation of the gradients of the production rate $\partial f/\partial bi$ with respect to the decision variables (buffer capacities) $b_i, i = 1,2,\ldots,n-1$. This solution method is derived from the optimisation approaches by the technique of gradient descent (Robbins and Monro 1951).

Consider the original simulation that consists in simulating P parts. Each iteration k is done by simulating L parts (where $L < P$), after which a new evaluation of the gradients is calculated by FPA. This evaluation determines the new search direction for the design solution. This search method requires, for each iteration k, the estimation of the projections' vectors of the gradients on the hyperplane $\sum_{i=1}^{n-1} b_i = B_{\max}$. Indeed, for a parameter b_i, the projected gradients allow to determine the best direction for improving the current solution (Robbins and Monro 1951). Thereafter, an iterative hill-climbing-type procedure is used. The objective is to evolve from the solution found towards a better solution using the gradient descent technique. The iteration $(k+1)$ improves, from one configuration to another, the parameters b_i^k, representing the buffer capacity b_i at the iteration k.

Hence:

$$b_i^{k+1} = b_i^k + a_k \cdot \left(\frac{\partial f}{\partial b_i} - \frac{1}{n-1} \sum_{i=1}^{n-1} \frac{\partial f}{\partial b_i} \right) \quad (3)$$

where,

- a_k is a numeric suite that verify the following conditions of convergence:

$$\lim_{k \to \infty} a_k = 0, \sum_{k=1}^{\infty} a_k = \infty, \sum_{k=1}^{\infty} a_k^2 < \infty \quad (4)$$

- The term $\left(\frac{\partial f}{\partial b_i} - \frac{1}{n-1} \sum_{i=1}^{n-1} \frac{\partial f}{\partial b_i} \right)$ in equation (3) represents the projection of the gradient $\frac{\partial f}{\partial b_i}$ on the hyperplane constraints $\sum_{i=1}^{n-1} b_i = B_{\max}$.

SA stops if it is not possible to improve the *PR* anymore with a new solution (the gradients' values become negligible), or when the maximum number of parts produced in the simulation is reached.

4. Development of the optimisation technique

4.1. Genetic algorithm (GA)

GA is a global optimisation technique that considers an optimisation problem as the environment where feasible solutions are the individuals living in that environment. A set of feasible solutions takes the place of a population of organisms. Every organism represents a possible valid solution to the problem. In general, a GA begins with a randomly generated set of individuals. Once it has been generated, the GA enters a main loop. After each iteration (generation), the population changes and a new population is produced by applying a number of stochastic operators to the previous population (Tomassini 1999).

We present below the components of the proposed GA, relying on the following notation.

- N is the number of individuals
- $P(t)$ is the current population
- $P(t+1)$ is the new population
- *MinBuffer* is the smallest value allowed
- *MaxBuffer* is the largest value allowed
- *PositionBuffer* is the rank of the buffer
- *GapStock* is a parameter that allows adjusting the difference between *MinBuffer* and *MaxBuffer*
- $U\{a,b\}$ is a function that randomly generates (with a uniform distribution) an integer in the set {a, b}
- $Int[c]$ is an operator that takes only the integer value of c

Initial population: the first step of the proposed GA consists of building an initial population using Algorithm 1. The initial population is composed of m different solutions (or configurations). Each configuration is generated randomly with the discrete uniform distribution. To satisfy Constraint (2), some buffer values are then increased or decreased in a well-distributed manner i.e. a repair procedure is performed to make the solution feasible. The repair procedure works as follows. It distributes, over the buffers, the value of a possible gap between (a) and (b), where (a) is the sum of the buffer capacities that are generated by the uniform distribution, and (b) is the total capacity B_{max} required by Constraint (2) ($\sum_{i=1}^{n-1} b_i = B_{\max}$). If the gap is negative as (a) < (b) (resp. positive as (a) > (b)), the repair is performed by sequentially increasing (resp. decreasing) by one unit the value of some b_i's, picked up randomly among the b_i's that are below (resp. above) the largest (resp. smallest) b_i, until a feasible solution is reached. To illustrate this procedure, we give an example of allocation of 30 buffers at four positions. Suppose that the uniform distribution gives the configuration (3, 8, 6, 10), which amounts to 27 (the gap is thus equal to –3). Therefore, to satisfy the constraint

of reaching 30, the repair procedure could increase by one unit the values of the first three stocks to finally give the following configuration (4, 9, 7, 10). The main objective of this phase is to have configurations that cover, in a non-biased manner, various regions of the solution space.

Algorithm 1. Generation of initial population of m individuals

While the number of individuals is not reached, **do**

For each individual, **do**

SumBuffer = 0 (initialisation of the sum of values of all the capacities allocated to the stocks)

MinBuffer = *Int*[B_{max} / (*n*-1)] – *GapStock* (*n*-1 denotes the size of the generated individual; *n* is the number of machines)

MaxBuffer = *Int*[B_{max} / (*n*-1)] + *GapStock*

For each position, **do**

PositionBuffer ~ *U*{*MinBuffer*, *MaxBuffer*} (position gives the rank of the buffer) *PositionBuffer* = *SumBuffer*+ *PositionBuffer*

$D = B_{\max}$ (if D ≠ 0, the repair procedure is applied)

If $D = 0$, the individual is already feasible

If $D > 0$, increase in a well-distributed way some values of the individual

If $D < 0$, reduce in a well-distributed way some values of the individual

Evaluation: the fitness value of each individual, namely *f*, represents the *PR* of the production line and is the result of the execution of a simulation model (Arena).

Solution encoding: each decision variable is associated with a gene. A set of genes constitutes a chromosome, and one or several chromosomes represent an individual (i.e. a solution). In this paper, there is one type of gene: buffer capacity. Thus, each individual consists of a unique chromosome $s = (b_1, b_2, \ldots, b_{n-1})$.

Selection: the tournament selection is used, which is a useful and robust selection mechanism commonly used by GAs (Miller and Goldberg 1995). It consists in first selecting randomly two individuals of the population, then in comparing their *PR*, and finally in selecting the winner for the next generation.

Crossover: two parent solutions of the current population interact and give two child solutions (offsprings) of the next population (for the next generation). The major reason that makes this crossover operator very suitable to the problem under consideration is that it assures feasibility of the offspring. Since both parents are feasible, both children must also be feasible (Matondang and Jambak 2010). The crossover phenomenon is essential because it allows GA to explore the search space efficiently. In this paper, we use the arithmetic crossover operator with a constraint criterion (Duman 2018). From a population of parents, two parent solutions s_1 and s_2 are first selected. Next, the two new offsprings s'_1 and s'_2 are generated by linear combination of s_1 and s_2 as follows:

$$s'_1 = \alpha \cdot s_1 + (1 - \alpha) \cdot s_2 \quad (5)$$

$$s'_2 = (1 - \alpha) \cdot s_1 + \alpha \cdot s_2 \quad (6)$$

where α is a random weighting coefficient picked uniformly in the interval [0, 1]. Note that in order to respect Constraint (2), a balanced adjustment for the arithmetic crossover is used. More precisely, some values of the offsprings are increased or decreased in a well-distributed manner in order to respect the total buffer space available for the whole production line (i.e. the repair procedure is performed). Moreover, since the buffer sizes must be integer, some values of the offsprings are modified to satisfy this condition. The former repair procedure and the latter rounding mechanism correspond here to the mutation operator.

The proposed GA is described in Algorithm 2.

Algorithm 2 GA Genetic Algorithm

Initialisation

Set $t = 0$

Generate randomly an initial population $P(t)$ of m individuals.

Calculate the production rate of each individual of $P(t)$ using the Arena simulation model.

While $P(t+1)$ does not contain N individuals, **do**

Repeat $N/2$ times the below process:

Select randomly two parents from $P(t)$ and apply the tournament selection based on the fitness *f*.

Clone the best-selected individual.

Apply arithmetic crossover operator between two parent solutions s_1 and s_2.

Insert the offsprings s'_1 and s'_2 in the new population $P(t+1)$.

Repeat the While loop as long as an overall stopping condition is not met

4.2. Finite perturbation analysis (FPA)

Discrete-event dynamic systems (DEDS), such as manufacturing systems, are dynamic asynchronous system where the state transitions are initiated by the occurrence of discrete events in the system at instants of time. The evolution in time of a DEDS is analyzed using its trajectory, denoted by *Nominal Trajectory* (NT), representing the events that affect the entities processed by the system. Assuming that a parameter of the system is perturbed in a trajectory following the occurrence of an event, the value of the parameter changes, and a new trajectory is obtained, denoted by *Perturbed Trajectory* (PT).

Perturbation analysis (PA) is an analytical technique that calculates the performance-measure sensitivity of

a DEDS with respect to system parameters, by analyzing its sample path (Ho 1987). PA allows obtaining the perturbed performance from a nominal (original) experiment or sample path without the need of doing a perturbed experiment. When the perturbations are finite (large amplitude), and where the changes of order of events in the trajectory do not influence the performance measure of the system, an extension of the PA must be introduced. It is called FPA or first order PA (Cao 1987). We can thus emit the hypothesis that once a perturbation is introduced, the type of future interactions that it can encounter and/or lead through system (machines, buffers and parts in the system) in the PT is statistically similar to that in the nominal one (Ho 1987). Hence, the construction and simulation of the PT is not necessary and only the NT will be used to define the system's response to a perturbation. This development is supported by specific experimental results of particular DEDS (Heidelberg et al. 1988).

The purpose of applying the perturbation analysis is to provide a prediction of an estimate of *PR* through the development of two sets of rules: the perturbation generation rules and the perturbation propagation. These rules are based on the prediction of future events and their respective durations after the introduction of perturbations in the NT by changing the capacity of some buffers. First, we consider the unitary cell shown in Figure 3 for the development of the FPA rules. Next, from the different results obtained, we can generalise them to the entire production line.

Consider the sequence of events of the NT of the machine M_i, denoted as $M_i(t)$, and that of M_{i+1}, denoted as $M_{i+1}(t)$, and the buffer level $b_i(t)$ of the Unitary cell (see Figure 4). The occurrence of a long-duration failure of M_{i+1} leads to an accumulation of parts in buffer b_i,

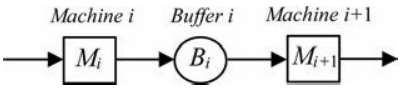

Figure 3. Unitary cell.

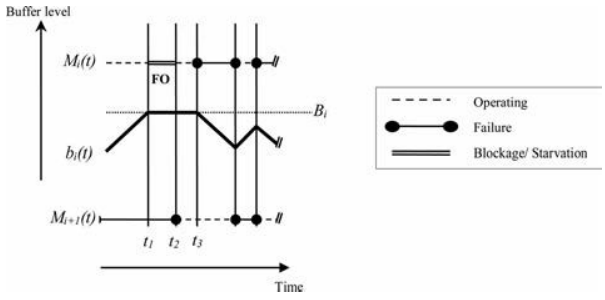

Figure 4. Nominal trajectory of a unitary cell.

supplied by M_i that is in operating order. At time t_1, $b_i(t)$ reaches its maximum capacity B_i and any part produced by M_i cannot be stored in the buffer. Although machine M_i remains operational, it becomes blocked (noted FO) until the instant t_2, when the machine M_{i+1} is finally repaired and becomes operational. Next, it begins to process parts from the intermediate stock b_i. Hence, this marks the end of the blocked period for machine M_i. At time t_3, M_i breaks down, allowing M_{i+1} to continue to process the parts already in the stock b_i. This explains the reduction in the level of b_i, which subsequently continues to vary with the different states of M_i and M_{i+1}. The blocked period of Figure 4 could be reduced if the buffer capacity of b_i is higher.

During the period $[t_1, t_2]$, the storage space remains saturated, as shown on the trajectory. This is due to our choice of representation of the total stock capacity in the model that takes into account the unit on M_i waiting to be placed in b_i. This unit is directly stored as soon as a place is released in b_i. Moreover, since the buffer is a discrete parameter, we should have represented the evolution of the level of b_i in steps, but for sake of clarity, we preferred a representation where the buffer level is considered as if it is a continuous variable.

4.2.1. Perturbation generation rules in a unitary cell

All the propositions hereafter are described in detail in (Cheikhrouhou 2001) and they form the core of the FPA.

Proposition 4.1: *Let $M_i(t)$ be a blocked machine (FO) during a period $[t_1, t_2]$ of the nominal trajectory. A perturbation (or local gain) of size Δt_i is created in the sequence of events of this machine, if a variation in the size of stock Δb_i is injected when M_i passes through the FO period (or earlier). The perturbed trajectory of M_i, denoted $M_i'(t)$, is specified as follows:*

$$M_i'(t) = M_i(t + \Delta t_i), \forall t > t_1 \tag{7}$$

$$\Delta t_i = t_i \cdot \Delta b_i \tag{8}$$

where t_i is the average service time of M_i, and Δt_i is the resultant perturbation in its trajectory. Figure 5(a) graphically reflects equations (7) and (8) for an eligible perturbation provoked by $\Delta b_i = 1$ and where the sequence of events $M_i(t)$ is translated in time by a value of Δt_i corresponding to the equivalent gain in production time.

Note that if the perturbation is not eligible for this interval, i.e. $(t_2 - t_1) < \Delta t_i$, the FO period would be completely removed from the perturbed trajectory. The sequence of events of $M_i$would be advanced by a period equal to $(t_2 - t_1)$. Moreover, the perturbation may be considered as a local gain because it represents a potential

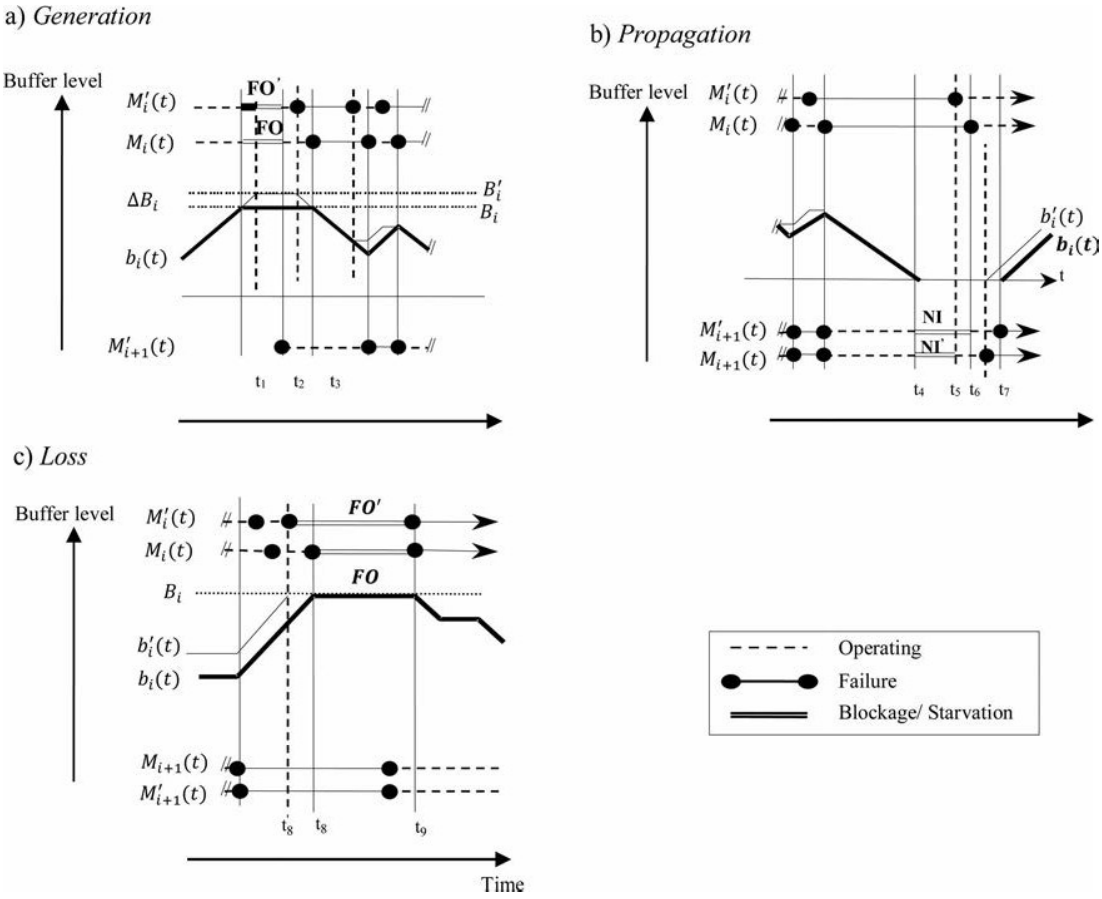

Figure 5. Generation, propagation and loss of perturbation in a unitary cell.

gain in the global execution time, if it is passed on to the global duration of the production.

4.2.2. Perturbation propagation rules in a unitary cell

Proposition 4.2: *Let* $[t_4, t_6]$ *be a time interval in which machine* M_{i+1} *of the unitary cell is starved (noted NI). The operation* $\Delta b_i = 1$ *does not change the input date of period* t_4 *between the nominal trajectory and the perturbed trajectory. Hence*

$$b_i(t_4) = b'_i(t_4) = 0 \tag{9}$$

where $b'_i(t)$ *is the buffer level in the PT. The same result can be formulated in the most general case with eligible perturbations* $\Delta b_i = p, \forall p \in \mathbb{N}$.

Proposition 4.3: Equation (7) *and the hypothesis of the existence of a period of famine (NI) (due to a failure for example) for* M_{i+1} *during the following period* $[t_4, t_6]$ *in the NT induces a reduction of this interval in the PT. The new time interval NI is* $[t_4, t_5]$*, calculated from* (10)*. The perturbation is therefore completely propagated from* M_i *to* M_{i+1}.

$$NI'[t_4, t_5] = FO[t_4, t_6] - \Delta t_i \tag{10}$$

$$M'_{i+1}(t) = M_{i+1}(t + \Delta t_i),\ \forall t > t_5 \tag{11}$$

The sequence of events of M_i denoted by $M'_i(t)$ in its perturbed trajectory M_{i+1} is represented in Figure 5(b). The value of the total propagated perturbation from $M_i$to M_{i+1} is calculated by:

$$\Delta t'_i = t_6 - t_5 = \Delta t_i \tag{12}$$

Proposition 4.4: *Suppose that the trajectory* $M_i(t)$ *is translated to the left by* Δt_i*, due to a particular gain (propagation of perturbation) at an earlier time (in the perturbed trajectory). Since* M_{i+1} *did not undergo any perturbation until that moment and* B_i *has not been modified, the occurrence of an event FO during* $[t_8, t_9]$ *in the nominal trajectory of* $M_i(t)$ *implies an increase of the FO period in the perturbed trajectory and consequently, the loss of the perturbation (see Figure 5(c)):*

$$FO'[t_8, t_9] = FO[t_8, t_9] + \Delta t_i \tag{13}$$

$$M'_i(t) = M_i(t), \forall t > t_8 \tag{14}$$

Note that if the duration of the blockage is considered very short, the same scheme is applicable with the same explanation. However, this period may disappear in the perturbed trajectory if the value of the perturbation is longer than the duration of the blockage.

4.3. Description of the FPA and SA algorithms

4.3.1. Notation

P	number of parts for each simulation
L	number of parts for each iteration
T	duration of a simulation
f	production rate in the nominal trajectory ($f = P/T$)
f^*	estimated production rate in the perturbed trajectory
$\partial f/\partial bi$	gradients of the production rate due to buffer decision variable b_i
N	number of iterations
b_i^k	buffer between the machine i and the machine $i+1$ at iteration k
Sum_i	accumulation of the gain time on machine i
A	regulation parameter for the convergence
Anint	an Arena operator that takes the near integer value
Abs	absolute value
Arg min	an arena operator that takes the minimum value

We assume that the simulation of a production line required to produce P parts is sufficiently long so that all possible events can occur during the simulation run. The PR is estimated by f, where T is the duration of the simulation. Then for a given i,

$$\frac{\partial f}{\partial b_i} = \frac{\partial\left(\frac{P}{T}\right)}{\partial b_i} = -\frac{f}{T}\cdot\frac{\partial T}{\partial b_i} \tag{15}$$

where $\partial T/\partial bi$ represents the total perturbation time affecting the line, due to the introduction of the perturbation Δb_i in the simulation. The value of $\partial T/\partial bi$ is calculated by applying the generation and propagation rules of the FPA mentioned above.

4.3.2. FPA algorithm

The presented FPA relies on (Suri 1989) and it is applicable only for DEDS flow-shop such as manufacturing systems or communication networks. FPA includes the detection of critical events in the nominal trajectory (such as the FO and the NI), and their respective durations. Therefore, we can predict the repercussion of a perturbation of *one unit capacity* of the stock associated with each relevant event. This repercussion concerns the persistence of the event in the perturbed trajectory and its duration. Algorithm 3 describes the FPA algorithm for the optimisation of buffer capacities. It works simultaneously with the simulation of the nominal trajectory in order to detect in time the events which make it possible to generate and propagate the perturbations, such as the beginning of a FO period or the end of a NI period.

Algorithm 3 FPA Finite Perturbation Analysis

Generation of perturbations

Step 1

For $i = 1$ to n-1, **do**: $Sum_{i+1} = 0$; Select b_i (initialisation of accumulators Sum_{i+1})

Step 2

For « $Mi{+}1(t)$ is FO for the first time », **do**: $B_i = B_i + \Delta B_i$; $Sum_i = t_i \cdot \Delta B_i$

Propagation of perturbation

Step 3

For any event « $Mi(t)$ is NI », **do**: $Sum_i = \max\{Sum_j + [NI], Sum_{i+1}\} - \max\{0, [NI]\}$

Step 4

For any event « $Mi{+}1(t)$ is FO », **do**: $Sum_i = \max\{Sum_{i+1} + [FO], Sum_i\} - \max\{0, [FO]\}$

Step 5

If $i = n$ (Last machine), **then:** $\frac{\partial f}{\partial b_i} = -\left(\frac{f}{T}\right)\cdot Sum_n$; $f^* = \frac{P}{T - Sum_n}$; Stop

Else Go to Step 1

4.3.3. Stochastic algorithm (SA)

The proposed SA is based on the Robbins-Monro procedure. It is an iterative procedure used to update the buffer capacities during the simulation run, i.e. if at each update a new calculation of the gradients (thus a launch of the FPA algorithm) is done, we need as many launches as the number of buffer updates. The total duration of simulation would be too long. To overcome this disadvantage, the solution is to develop a single simulation optimisation (mono-run). FPA is then integrated into SA (see Algorithm 4) that is used to reach a potential optimal/near-optimal solution. It is based on the nominal trajectory that is available to update the configuration settings (here, the buffer capacities) for each production of a number L of parts. Step 3 presents the modified procedure of Robbins-Monro that uses the technique of gradient calculation based on FPA. A perturbation is indeed an incrementation of one unit of the capacity of a buffer that shows blockage events during his nominal trajectory. This incrementation is virtual and used only to evaluate the Production Rate gradient thanks to the propagation rules of FPA. Actually, the values of the buffers capacities are updated during the optimisation using SA and not through their incrementation by one unit. Moreover, each update of the buffer capacities at an iteration (k+1)

respects the main constraint $\sum_{i=1}^{n-1} b_i^{k+1} = \sum_{i=1}^{n-1} b_i^k = B_{\max}, \forall n$, which leads to the fact that, at any iteration, the algorithm keeps the allocation of the same global capacity on all line buffers. Step 4 is a precaution case where some updated values of b_i^{k+1} are negatives. We choose a_k as numeric suites of type A/k, where A is a constant randomly determined by choosing an initial value. The adjustment is done through the calculation of the projected gradient values, so that the step size between the iterations is consistent. In other words, the value of $b_i$after each iteration is in the same order of magnitude.

Algorithm 4. SA Stochastic Algorithm

Initialisation	$k = 1$
Step 1	Choose the initial values of b_i^k, for $i = 1$to n-1
Step 2	Simulate for b_i^k L parts ($L < P$) Estimate $\frac{\partial f}{\partial b_i}$ by using FPA
Step 3	$b_i^{k+1} = b_i^k + \left(\frac{A}{N}\right) \cdot \left(\frac{\partial f}{\partial b_i} - \frac{1}{n-1} \sum_{i=1}^{n} \frac{\partial f}{\partial b_i}\right)$
Step 4 If	$b_i^{k+1} \leq 0$ **then** $b_p^k = \text{Arg min } b_i^k$ $b_p^{k+1} = \text{Arg min } b_i^{k+1}$ $a \sim U(0,1)$ $f = \left\lvert \frac{a \cdot b_p^k}{b_p^{k+1} - b_p^k} \right\rvert$ $b_i^{k+1} = b_i^k + f \cdot \left(\frac{A}{N}\right) \cdot \left(\frac{\partial f}{\partial b_i} - \frac{1}{n-1} \sum_{i=1}^{n} \frac{\partial f}{\partial b_i}\right)$
Step 5 If	$\lvert b_p^{k+1} - b_p^k \rvert \leq \epsilon$ **then** $b_i = Anint(b_i^k)$; STOP
If	P parts are simulated **then** STOP
Else	$k = k + 1$and go to step 2

Since we are dealing with discrete values, a discretisation of stock values would be necessary. The idea is to consider the capacities of the stocks as continuous variables until the end of simulation of L parts. After that, the algorithm transforms the value of the parameter to a discrete value (Step 5). The parameter ϵ represents a trade-off between the simulation length and the accuracy of the estimated parameter, for which a value of 0.0001 is taken in this work. The advantage of this stopping criterion is that it takes into account the variations of the all the $b_i$variables simultaneously. Therefore, the algorithm stops when the difference between the values of b_i from an update to another becomes negligible or when the maximum number of parts produced in the simulation is reached.

Note that by changing the values of $b_i$during the single-run simulation, transient effects are introduced into the system and the *PR* values and gradients thus contain biases related to these effects. The algorithm proposed at this stage is therefore considered as a heuristic.

5. Numerical experiments

In Subsection 5.1, the parameters of the proposed method are identified. Numerical comparisons of our methods are performed in Subsection 5.2 with respect to the literature. Four sets of instances are considered, involving 3, 5, 10 and 20 machines in the production line. Finally, Section 5.3 demonstrates the benefit of combining GA with FPA when compared to GA or FPA only. In each table, the best results are always highlighted in bold. The *Arena* simulation language V14.0 is used to develop the models and the algorithms are implemented in Java. The experiments are performed on a PC with a 1.9-GHz Core (TM) i5CPU processor with 8 GB of RAM.

Termination criterion: the algorithm stops when the production rate obtained at a given generation is lower than the best production rate obtained so far (only valid after the 10th generation), or when the maximum number of 20 generations is reached.

5.1. Determination of the parameters of the methods

On the one hand, we have to determine the number R of replications. A large R means a long computational time, and conversely a simulation with a small R can lead to a biased solution. It is therefore necessary to determine when the method converges into fewer replications. Several test cases for different sizes of production lines are conducted to predict the evolution of the average *PR* with respect to R.

Figure 6 shows four instances involving 3, 5, 10 and 20 machines, with R ranging from 1 to 50. It contains: (A) a Tukey's boxplot to reflect the distributions of the *PRs*; (B) a graph of the average *PR* for different values of R. From the graphs, we can see that the average *PR* is stable from $R = 30$ and above. This coincides with the law of large numbers, which states that the average of a large independent random variable approximately converges to their expectation. Generally, this approximation can be used when the size of the sample is greater than 30. Therefore, in this paper, $R = 30$ replications will be used.

On the other hand, we have to determine the operator parameters of GA, as they play a crucial role in the convergence of the algorithm. Hence, for a given initial population, the convergence can be obtained faster while obtaining the best possible solution if the operators are well designed (Pavai and Geetha 2019). After preliminary experiments, we have set 20 generations and 30 individuals for GA. For the selection operator, we have tested *Random Selection* (RS) and *Tournament Selection* (TS). For the crossover operator, we have tested *Single-Point Crossover* (SPC) and *Arithmetic Crossover* (AC).

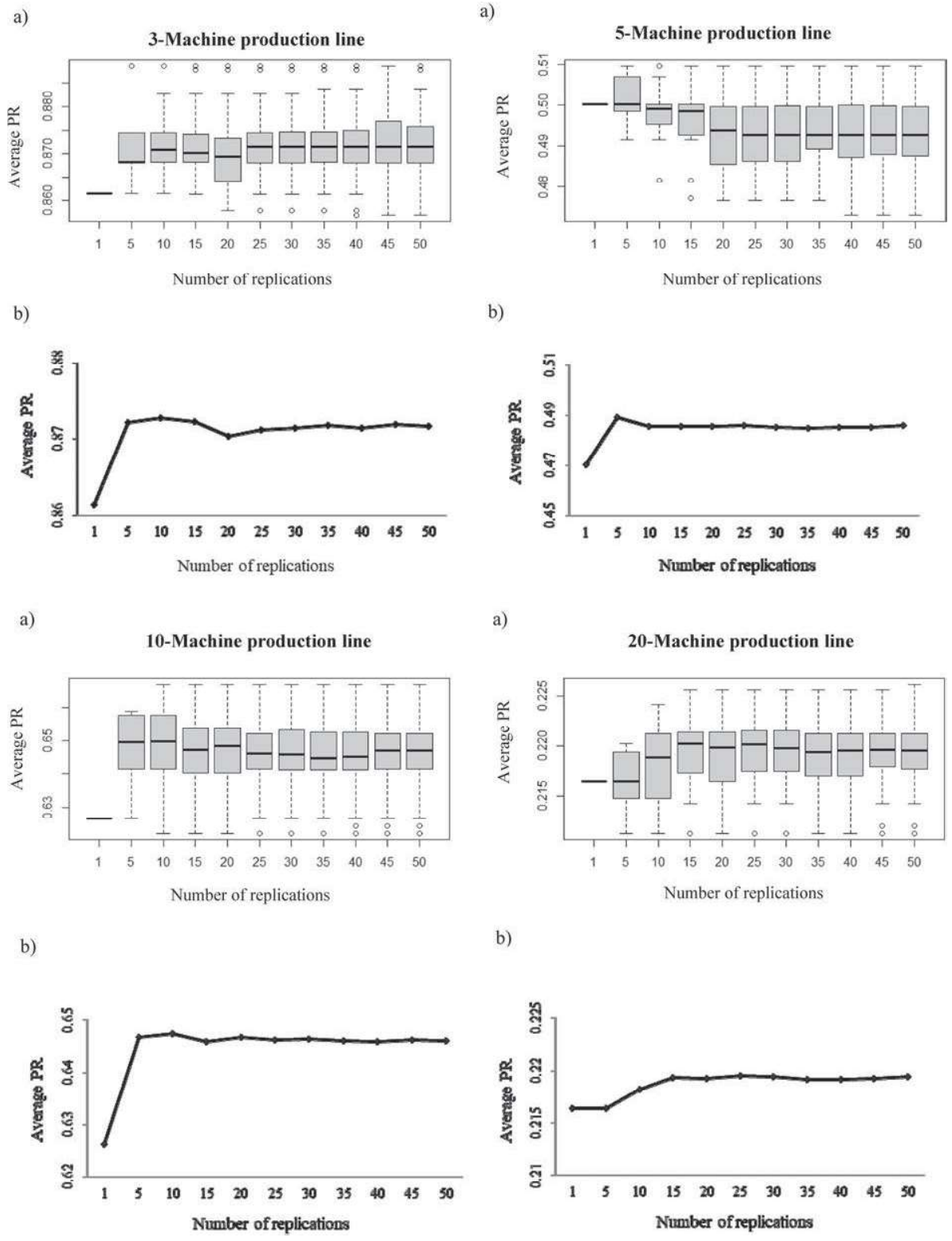

Figure 6. Evolution of production rate with different numbers of replications.

Table 1. Identification of parameters of GA.

Number of machines	B_{max}	Average *PR*			
		RS/SPC	RS/AC	TS/SPC	TS/AC
3	20	0.871783	0.871783	0.871783	**0.871783**
5	31	0.494708	0.494708	0.494789	**0.496413**
10	270	0.647766	0.646903	0.645671	**0.649204**
20	100	0.351846	0.354975	0.354220	**0.358180**
40	200	0.436278	0.438015	0.436319	**0.438583**

The computational results are summarised in Table 1. The first two columns in this table depict the size of the production line (i.e. the number of machines and the total buffer size available). The next columns show the average *PR* for the following combinations of operators (RS-SPC), (RS-AC), (TS-SPC), and (TS-AC), respectively. One can see that for the case with 3 machines, all the PR values are equal. However, when the number of machines increases, the best results are obtained when using the (TS-AC) combination.

5.2. Comparison of the proposed method with respect to the literature

GA-FPA is compared with different methods from the literature (see Table 2). All the runs are conducted under the same experimental conditions. More precisely, in all sets of problems, it is assumed that the service times of each machine is one time unit, the machines are subject to breakdown, and the failure and repair times follow a geometric distribution with a probability of $p_i$and r_i, respectively (these values will be presented later).

5.2.1. Results for a 3-machine production line

Consider a production line composed of 3 machines and 2 buffers. The information on the machines are given in Table 3. This example is proposed by Gershwin and Schor (2000), and used as a comparison benchmark by Massim et al. (2012). In their study, Gershwin and Schor (2000) do not mention explicitly the buffer configuration found and they only give an idea on how production rate varies with respect to buffer sizes. This is the reason why GA-FPA is compared only with the value proposed by Massim et al. (2012). The total buffer capacities to be allocated among the line is 20. Table 3 shows that GA-FPA obtains higher *PR*.

5.2.2. Results for a 5-machine production line

This example was initially proposed by Ho, Eyler, and Chien (1979) and used in different papers as a benchmarking case (Gershwin and Schor 2000; Massim et al. 2012; Demir, Tunali, and Eliiyi 2011; Kose and Kilincci 2015). The total buffer size is 31, and the parameters of the five machines are presented in Table 4. For this example, the average production rate is estimated by simulating 100,000 parts with 50 replications. MTTR and MTBF denote the mean time to repair and the mean time between failures, respectively. Table 4 presents the buffer sizes configuration and the corresponding average

Table 2. State-of-the-art methods for buffer allocation.

Authors	Objective	System/method	Proposed method
Ho, Eyler, and Chien (1979)	Max *PR*	Simulation	Gradient technique based on perturbation analysis
Gershwin and Schor (2000)	Max PRMin (work in process, total buffer size)	Decomposition	Gradient search
Shi and Men (2003)	Max PR	Decomposition	Tabu search and nested partitions
Nahas, Ait-Kadi, and Nourelfath (2006)	Max PR	Decomposition	Degraded ceiling algorithm and simulated annealing
Demir, Tunali, and Eliiyi (2011)	Max *PR*Min (work in process, total buffer size)	Decomposition	Adaptive tabu search
Massim et al. (2012)	Max (*PR*, Profit)	Decomposition	Artificial immune algorithm
Kose and Kilincci (2015)	Max *PR*	Simulation	Genetic algorithm and simulated annealing

Table 3. Parameters and results obtained for a 3-machine production line.

Machine	1	2	3
r_i	0.35	0.15	0.4
p_i	0.037	0.015	0.02
Authors	B_{max}	Buffer size configuration	Average *PR*
Massim et al. (2012)	20	14 ... 6	0.86799
GA-FPA	20	13 ... 7	0.87178

Table 4. Parameters and results obtained for a 5-machine production line.

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

Authors	B_{max}	Buffer size configuration	Average *PR*
Ho, Eyler, and Chien (1979)	31	5 ... 11 ... 8 ... 7	0.4914
Gershwin and Schor (2000)	31	7 ... 10 ... 10 ... 4	0.4943
Demir, Tunali, and Eliiyi (2011)	31	7 ... 10 ... 10 ... 4	0.4943
Massim et al. (2012)	31	7 ... 10 ... 10 ... 4	0.4943
Kose and Kilincci (2015)	31	7 ... 10 ... 10 ... 4	0.4943
GA-FPA	31	7 ... 11 ... 9 ... 4	0.4948

Table 5. Parameters and results obtained for a 10-machine production line.

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 $= 1/p_i$	20	30	22	22	25	40	23	30	45	20

Authors	B_{max}	Buffer size configuration	Average *PR*
Nahas, Ait-Kadi, and Nourelfath (2006)	270	14 ... 19 ... 30 ... 54 ... 45 ... 27 ... 23 ... 24 ... 34	0.64135
Demir, Tunali, and Eliiyi (2011)	270	14 ... 19 ... 30 ... 54 ... 45 ... 27 ... 23 ... 24 ... 34	0.64135
Massim et al. (2012)	270	14 ... 19 ... 30 ... 52 ... 47 ... 27 ... 23 ... 24 ... 34	0.64139
Kose and Kilincci (2015)	270	7 ... 16 ... 48 ... 61 ... 24 ... 41 ... 20 ... 34 ... 19	0.63016
GA-FPA	270	19 ... 23 ... 24 ... 45 ... 43 ... 34 ... 22 ... 29 ... 31	0.64920

production rates obtained by the different approaches. GA-FPA obtains a slightly better production rate with the buffer size configuration (7, 11, 9, 4), which has the same allocation at the extremities of the line as Kose and Kilincci (2015), Demir, Tunali, and Eliiyi (2011), Massim et al. (2012) and Gershwin and Schor (2000). The allocation of small capacities at the beginning and at the end of the line, and important sizes in the middle, is likely to be efficient for facilitating the passage of the parts and thus bypassing a possible congestion of the line. Again, we can see that GA-FPA obtains the best results.

5.2.3. Results for a 10-machine production line

This instance is proposed by Nahas, Ait-Kadi, and Nourelfath (2006) and used in (Massim et al. 2012; Demir, Tunali, and Eliiyi 2011; Kose and Kilincci 2015). The dataset is shown in Table 5. The total buffer size to be allocated is 270. As shown in Table 5, Demir, Tunali, and Eliiyi (2011) and Nahas, Ait-Kadi, and Nourelfath (2006) obtain the same buffer configuration (14, 19, 30, 54, 45, 27, 23, 24, 34) with an average *PR* of 0.64135. Massim et al. (2012) report a slightly better solution of 0.64139 with an almost similar distribution of buffers (14, 19, 30, 52, 47, 27, 23, 24, 34). GA-FPA generates the best *PR* with a different buffer allocation (19, 23, 24, 45, 43, 34, 22, 29, 31). It can be noted that when the machines are balanced, all the methods generate, in most cases, distributions of buffers capacities with more storage areas allocated in the middle of the line. This buffer allocation is likely to avoid blockages and to facilitate parts flowing downstream.

5.2.4. Results for a 20-machine production line

The three first columns in Table 6 present the parameters of the production line. The fourth and fifth columns give the results obtained by Demir, Tunali, and Eliiyi (2011) and Kose and Kilincci (2015), respectively. The last column gives the results obtained by GA-FPA, which outperforms the other methods for 8 out of 9 instances. The case where our method does not outperform the existing methods is due to the lower value of the Production Rate of one of the replications where the PR value is 0.20357. In this case, the nominal trajectory in the simulation presents a high number of starving periods (NI) which increased the total production time to produce the needed number of parts, thus decreasing the production rate.

Table 6. Results obtained for a 20-machine production line.

Parameters			Average *PR*		
p_i	r_i	$B_{\max}$	Demir, Tunali, and Eliiyi (2011)	Kose and Kilincci (2015)	GA-FPA
0.1	0.1	100	**0.234191**	0.226499	0.223559
0.2	0.2	100	0.297025	0.302448	**0.305203**
0.3	0.3	100	0.334450	0.349517	**0.354811**
0.4	0.4	100	0.359637	0.354299	**0.394316**
0.5	0.5	100	0.377895	0.382401	**0.422175**
0.6	0.6	100	0.391846	0.397120	**0.443916**
0.7	0.7	100	0.402760	0.446904	**0.461407**
0.8	0.8	100	0.411638	0.450032	**0.476649**
0.9	0.9	100	0.419018	0.459111	**0.487760**

Table 7. Parameters and results for a 5-machine production line.

Parameters			Average *PR*			CPU (s)		
p_i	r_i	$B_{\max}$	GA	FPA	GA-FPA	GA	FPA	GA-FPA
0.1	0.1	50	0.344618	0.341419	**0.344715**	797	**363**	468
0.2	0.2	50	**0.406667**	0.405393	**0.406667**	235	881	**226**
0.3	0.3	50	0.437662	0.436675	**0.438030**	475	377	**320**
0.4	0.4	50	0.457296	0.456333	**0.457545**	581	387	**251**
0.5	0.5	50	0.469631	0.468800	**0.470165**	686	**338**	455
0.6	0.6	50	0.479729	0.479060	**0.479993**	**744**	845	762
0.7	0.7	50	0.486405	0.485869	**0.486442**	597	624	**359**
0.8	0.8	50	0.491948	0.491634	**0.492158**	784	753	**588**
0.9	0.9	50	**0.495934**	0.495813	**0.495934**	367	**287**	310

Table 8. Parameters and results for a 10-machine production line.

Parameters			Average *PR*			CPU (s)		
p_i	r_i	FPA	GA-FPA	FPA	GA-FPA	GA		
0.1	0.1	100	0.333260	0.330340	**0.335682**	1435	2337	**1122**
0.2	0.2	100	0.422943	0.416035	**0.423577**	1899	1841	**1177**
0.3	0.3	100	0.475451	0.468834	**0.475533**	2708	2067	**1941**
0.4	0.4	100	0.505700	0.502103	**0.506404**	1563	**1486**	1645
0.5	0.5	100	0.524021	0.520809	**0.524761**	1573	1304	**1035**
0.6	0.6	100	0.533112	0.527433	**0.533875**	1559	1731	**1140**
0.7	0.7	100	0.531353	0.528109	**0.532405**	1603	**1243**	1270
0.8	0.8	100	0.524930	0.524174	**0.526177**	1051	2528	**950**
0.9	0.9	100	0.515859	0.514694	**0.516297**	1940	**1333**	1368

5.3. Benefit of combining GA and FPA

In order to measure the benefit of combining GA with FPA in a single method GA-FPA, an idea is to compare GA-FPA with its components GA and FPA considered alone. Results are given for instances with 5, 10 and 20 machines, in Tables 7–9, respectively. In each table, the data is first given (column 1–3), followed by the average *PR* for GA, FPA and GA-FPA (column 4–6), and finally, computing times (to reach the best encountered solutions) are provided in seconds at the end (column 7–9). The instances are characterised as follows: the service time of each machine is one time unit; the failure and repair times follow a geometric distribution with a probability of $p_i$and r_i, respectively; all machines are identical and have the same failure and repair probabilities, ranging from $p_i = r_i = 0.1$ to $p_i = r_i = 0.9$ with increments of 0.1. The total buffer size to be allocated is 50, 100 and 200 for the instances with 5, 10 and 20 machines, respectively.

On the one hand, we can easily observe that regarding quality, GA-FPA obtains the best *PA* results for each of the 27 instances (note that for two instances with 5 machines, GA obtains equivalent solutions). On the other hand, regarding speed, GA-FPA is the quickest method for 13 out of 27 instances. It seems that for the larger instances with 20 machines, GA-FPA need more time to converge to its best solutions. The success of GA-FPA can be explained by the fact that the GA component can quickly identify promising regions of the solutions space (exploration ability), whereas the FPA component can intensify the search in such regions (exploitation ability).

5.4. Sensitivity analysis

When all machines of the production line are identical (the same service times and the same probabilities of failure and repair), all the methods (our method as well as the methods with which we made a comparison) generate, in most cases, allocation of buffers capacities with more storage areas for the middle of the line. The purpose of the sensitivity analysis proposed is to determine whether this allocation is still respected in the case when the production line is unbalanced (different service times and parameters) and to determine, otherwise, whether there is a specific buffer allocation pattern.

Three series of experiments were carried out and a description of each of the test cases is presented. In all cases, the *PR* of the buffer allocations obtained by the GA-FPA approach is estimated by simulating 10,000 parts for 30 replications.

Table 9. Parameters and results for a 20-machine line.

Parameters			Average *PR*			CPU (s)		
p_i	r_i	B_{max}	GA	FPA	GA-FPA	GA	FPA	GA-FPA
0.1	0.1	200	0.286791	0.275098	**0.286923**	2559	4586	**2293**
0.2	0.2	200	0.364922	0.352133	**0.367650**	4880	**2295**	3032
0.3	0.3	200	0.408896	0.399193	**0.409025**	3261	**1890**	2287
0.4	0.4	200	0.435904	0.430023	**0.436041**	3881	**3577**	3950
0.5	0.5	200	0.453522	0.446965	**0.453614**	**4430**	6049	4620
0.6	0.6	200	0.466875	0.459231	**0.467263**	3616	**2105**	2340
0.7	0.7	200	0.476955	0.473205	**0.477123**	**2212**	6861	3489
0.8	0.8	200	0.484888	0.483725	**0.484942**	3934	3601	**3295**
0.9	0.9	200	0.491401	0.491431	**0.491442**	**1367**	5287	1930

Table 10. Parameters and results for set of experiments 1 for a 5-machine line.

Case	Service times	Buffer size configuration
1	$t_1 = 3, t_2 = t_3 = t_4 = t_5 = 1$	10 ... 9 ... 6 ... 5
2	$t_2 = 3, t_1 = t_3 = t_4 = t_5 = 1$	7 ... 11 ... 7 ... 5
3	$t_3 = 3, t_1 = t_2 = t_4 = t_5 = 1$	6 ... 9 ... 11 ... 4
4	$t_4 = 3, t_1 = t_2 = t_3 = t_5 = 1$	4 ... 9 ... 7 ... 10
5	$t_5 = 3, t_1 = t_2 = t_3 = t_4 = 1$	6 ... 6 ... 8 ... 10

Table 11. Parameters and results for set of experiments 2 for a 5-machine line.

Case	Repair rates	Buffer size configuration
1	$1/r_1 = 30, 1/r_2 = 1/r_3 = 1/r_4 = 1/r_5 = 10$	5 ... 7 ... 12 ... 6
2	$1/r_2 = 30, 1/r_1 = 1/r_3 = 1/r_4 = 1/r_5 = 10$	6 ... 8 ... 10 ... 6
3	$1/r_3 = 30, 1/r_1 = 1/r_2 = 1/r_4 = 1/r_5 = 10$	5 ... 9 ... 10 ... 6
4	$1/r_4 = 30, 1/r_1 = 1/r_2 = 1/r_3 = 1/r_5 = 10$	7 ... 11 ... 6 ... 6
5	$1/r_5 = 30, 1/r_1 = 1/r_2 = 1/r_3 = 1/r_4 = 10$	7 ... 11 ... 8 ... 4

Table 12. Parameters and results for set of experiments 3 for a 5-machine line.

Case	Failure rates	Buffer size configuration
1	$1/p_1 = 100, 1/p_2 = 1/p_3 = 1/p_4 = 1/p_5 = 300$	10 ... 9 ... 6 ... 5
2	$1/p_2 = 100, 1/p_1 = 1/p_3 = 1/p_4 = 1/p_5 = 300$	7 ... 11 ... 7 ... 5
3	$1/p_3 = 100, 1/p_1 = 1/p_2 = 1/p_4 = 1/p_5 = 300$	6 ... 9 ... 11 ... 4
4	$1/p_4 = 100, 1/p_1 = 1/p_2 = 1/p_3 = 1/p_5 = 300$	4 ... 9 ... 7 ... 10
5	$1/p_5 = 100, 1/p_1 = 1/p_2 = 1/p_3 = 1/p_4 = 300$	6 ... 6 ... 8 ... 10

Experiment set 1

In this case, we consider a 5-machine unbalanced production line. The total buffer size is set to 30. The different service times of the five machines are presented in the first column of Table 10, where the service time of one machine is, in each case, bigger than the other machines. The machines are subject to breakdown, and the failure and repair times follow geometric distributions with $p_i = 1/300$, and $r_i, = 1/10$ respectively.

Experiment set 2

We examine in this case the buffer allocation in a 5-machine balanced production line. It is assumed that the service time of every machine is one time unit, the failure times follow a geometric distribution with a probability

$p_i = 1/200$. The repair rates of the five machines are different, and presented in the first column of Table 11. In each of the 5 cases, just one machine needs less time to repair compared to the others machines.

Experiment set 3

This test examines the buffer allocation in a 5-machine balanced production line. The service times of each machine is one time unit and the repair rates of all machines are set to $r_i = 1/10$. The different failure rates of the five machines are presented in the first column of Table 12. In each case, just one machine has a high probability of breaking down compared to the other machines.

The results of the experiment set 1 (see Table 10) show that a larger stock size should be allocated to the input stock of the slower machine. This would allow the fastest upstream machines to continue to be operational and thus avoid any possible blockage.

The results of the experiment set 2 (see Table 11) show that in the case where the repair rates are different, the allocation found is comparable with the one in the case of balanced production lines, where a high accumulation of intermediate stocks is required. This buffer allocation is likely to avoid blockages and to facilitate parts flowing downstream.

From the results of the experiment set 3 (see Table 12), a configuration tends to persist, where the stock after the machine with a larger MTBF would require larger storage capacities. This could be explained by the need to facilitate parts flowing along the line, and thus prevent the upstream machines from being blocked and the downstream machines from starving.

6. Conclusion

This paper proposes a quick and efficient hybrid simulation-optimisation approach combining the genetic algorithm (GA) and the finite perturbation analysis (FPA) for the buffer allocation problem (BAP) of serial production lines with unreliable machines (the capacity of each buffer has to be decided, without exceeding the total available capacity for the production line). The heart of the proposed method is a stochastic algorithm based on the gradient estimates by FPA, complemented by GA for finding its input solutions. The role of GA is to approach quickly the optimal/near-optimal solution-space regions (thanks to its diversification and exploration ability), whereas the role of FPA is to investigate deeply such regions to find better solutions (thanks to its intensification and exploitation ability). An advantage of the proposed approach is to use a single simulation only. This is achieved because FPA can evaluate gradients of steady production rates at regular intervals during the simulation. The performed experiments show on the one hand that GA-FPA outperforms the state-of-the-art methods from the literature on instances where the number of machines varies from 3 to 20 machines. On the other hand, GA-FPA outperforms both GA solely and FPA solely with respect to solution quality, and often with respect to speed (i.e. the time needed to find its best solutions).

Various research directions could be derived from this work. First, GA-FPA could be extended to other types of production systems with different features, such as assembly/disassembly systems or transportation systems. Next, due to advances in manufacturing technologies and the rise of Industry 4.0, new solutions are opening up to improve the efficiencies and the production rates of these systems. The emergence of intelligent and interconnected collaborative industrial robots and storage sensors allows to monitoring the inventory levels and the different operations achieved by the robots, including material transfer between the different robots and operators in the workshop. This opens an avenue of research in the design of production lines. In fact, designing production lines would be not only allocating storage capacities on the different buffers, but it extends to the allocation of service times over the different robots and humans that are in charge of executing the operations on the line. Our design approach, based on optimisation using simulation, can be interesting to tackle such problems. In that case, the GA-FPA algorithms will be used to allocate, simultaneously, buffer capacities and robot/operator service times in a single simulation-optimisation run.

Disclosure statement

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

Notes on contributors

Khelil Kassoul obtained his electromechanical engineering degree from the *University of Boumerdes* in Algeria. Afterward, he continued his studies in Paris for an advanced degree in Electrical Engineering and worked as an associate member in a European project LHC (*Large Hadron Collider*) which is the largest and most powerful particle accelerator in the world at CERN (*European Organisation for Nuclear Research*). Holder of a postgraduate degree from EPFL (*Swiss Federal Institute of Technology* in Lausanne) in electrical engineering, he has been a professor of mathematics and physics for Swiss maturity and Baccalaureat classes since 2003. He is currently a PhD candidate at the *Geneva School of Economics and Management* of the *University of Geneva*. His research interests include areas related to applied mathematics, optimisation, design, simulation, artificial intelligence, and inventory management.

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 optimisation problems, with applications mainly in transportation, scheduling, production, inventory management, network design, and supply chain management. 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 120 publications (papers in professional journals, proceedings of conferences, and book chapters). He has had research activities with 26 Universities in Europe and America, as well as with 24 private companies.

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.

ORCID

Naoufel Cheikhrouhou http://orcid.org/0000-0003-2497-2528

References

Amiri, M., and A. Mohtashami. 2011. "Buffer Allocation in Unreliable Production Lines Based on Design of Experiments, Simulation, and Genetic Algorithm." *International Journal of Advanced Manufacturing Technology* 62 (1-4): 371–383.

Battini, D., A. Persona, and A. Regattieri. 2009. "Buffer Size Design Linked to Reliability Performance: A Simulative Study." *Computers & Industrial Engineering* 56 (4): 1633–1641.

Bierlaire, M., M. Thémans, and N. Zufferey. 2010. "A Heuristic for Nonlinear Global Optimization." *INFORMS Journal on Computing* 22 (1): 59–70.

Can, B., and C. Heavey. 2011. "Comparison of Experimental Designs for Simulation-Based Symbolic Regression of Manufacturing Systems." *Computers & Industrial Engineering* 61 (3): 447–462.

Cao, X. 1987. "First Order Perturbation Analysis of a Simple Multiclass Finite Source Queue." *Performance Evaluation* 7: 31–41.

Cheikhrouhou, N. 2001. "Apport de l'Analyse de Perturbation pour le Dimensionnement et le Pilotage des Systèmes de Production." PhD thesis, Grenoble Institute of Technology. https://www.theses.fr/2001INPG0001.

Cheikhrouhou, N., J.-L. Paris, and H. Pierreval. 2002. "Une Méthode de Dimensionnement de Lignes de Transfert via L'analyse de Perturbation Finie." *Journal Européen des Systèmes Automatisés* 36 (2): 199–221.

Costa, A., A. Alfieri, A. Matta, and S. Fichera. 2015. "A Parallel Tabu Search for Solving the Primal Buffer Allocation Problem in Serial Production Systems." *Computers & Operations Research* 64: 97–112.

Cruz, F. R. B., A. R. Duarte, and G. L. Souza. 2018. "Multiobjective Performance Improvements of General Finite Single-Server Queueing Networks." *Journal of Heuristics* 24 (5): 757–781.

Cruz, F. R. B., G. Kendall, L. While, A. R. Duarte, and N. L. C. Brito. 2012. "Throughput Maximization of Queueing Networks with Simultaneous Minimization of Service Rates and Buffers." *Mathematical Problems in Engineering* 2012: 1–19.

Cruz, F. R. B., T. Van Woensel, and J. M. Smith. 2010. "Buffer and Throughput Trade-Offs in M/G/1/K Queuing Networks: A Bicriteria Approach." *International Journal of Production Economics* 125: 224–234.

Demir, L., S. Tunali, and D. T. Eliiyi. 2011. "An Adaptive Tabu Search Approach for Buffer Allocation Problem in Unreliable non-Homogenous Production Lines." *Computers &Operations Research* 39 (7): 1477–1486.

Demir, L., S. Tunali, and D. T. Eliiyi. 2014. "The State of the Art on Buffer Allocation Problem: A Comprehensive Survey." *Journal of Intelligent Manufacturing* 25 (3): 371–392.

Diamantidis, A., J.-H. Lee, C. T. Papadopoulos, J. Li, and C. Heavey. 2019. "Performance Evaluation of Flow Lines with Non-Identical and Unreliable Parallel Machines and Finite Buffers." *International Journal of Production Research* 58 (13): 1–24.

Diamantidis, A., and C. T. Papadopoulos. 2009. "Exact Analysis of a Two-Workstation One-Buffer Flow Line with Parallel Unreliable Machines." *European Journal of Operational Research* 197 (2): 572–580.

Dolgui, A., A. Eremeev, A. Kolokolov, and V. Sigaev. 2002. "A Genetic Algorithm for the Allocation of Buffer Storage Capacities in a Production Line with Unreliable Machines." *Journal of Mathematical Modeling and Algorithms* 1: 89–104.

Dolgui, A., A. Eremeev, and V. Sigaev. 2007. "HBBA: Hybrid Algorithm for Buffer Allocation in Tandem Production Lines." *Journal of Intelligent Manufacturing* 18 (3): 411–420.

Duman, S. 2018. "A Modified Moth Swarm Algorithm Based on an Arithmetic Crossover for Constrained Optimization and Optimal Power Flow Problems." *IEEE Access* 6: 45394–45416.

Fu, M. C. 1994. "Optimization Via Simulation: A Review." *Annals of Operations Research* 53: 199–248.

Gendreau, M., and J.-Y. Potvin. 2019. "Handbook of Metaheuristics." *International Series in Operations Research and Management Science*. Springer International Publishing.

Gershwin, S. B., and J. E. Schor. 2000. "Efficient Algorithms for Buffer Space Allocation." *Annals of Operations Research* 93: 117–144.

Gürkan, G. 2000. "Simulation Optimization of Buffer Allocations in Production Lines with Unreliable Machines." *Annals of Operations Research* 93: 177–216.

Heidelberg, P., X. Cao, M. Zananis, and R. Suri. 1988. "Convergence Properties of Infinitesimal Perturbation Analysis Estimates." *Management Science* 34 (11): 1281–1302.

Hillier, F. S., and K. C. So. 1991. "The Effect of Machine Breakdowns and Interstage Storage on the Performance of Production Line Systems." *International Journal of Production Research* 29 (10): 2043–2055.

Ho, Y.-C. 1987. "Performance Evaluation and Perturbation Analysis of Discrete Event Dynamic Systems." *IEEE Transactions on Automatic Control* 32: 563–572.

Ho, Y. C., M. A. Eyler, and T. T. Chien. 1979. "A Gradient Technique for General Buffer Storage Design in a Production Line." *International Journal of Production Research* 17 (6): 557–580.

Kose, S. Y., and O. Kilincci. 2015. "Hybrid Approach for Buffer Allocation in Open Serial Production Lines." *Computers & Operations Research* 60: 67–78.

Kose, S. Y., and O. Kilincci. 2020. "A Multi-Objective Hybrid Evolutionary Approach for Buffer Allocation in Open Serial Production Lines." *Journal of Intelligent Manufacturing* 31: 33–51. https://doi.org/10.1007/s10845-018-1435-6.

Lee, H.-T., S. K. Chen, and S. Shunder. 2009. "A Meta-Heuristic Approach to Buffer Allocation in Production Line." *Journal of Creative Communication and Innovative Technology* 38 (1): 167–178.

Li, J., D. E. Blumenfeld, N. Huang, and J. M. Alden. 2009. "Throughput Analysis of Production Systems: Recent Advances and Future Topics." *International Journal of Production Research* 47 (14): 3823–2851.

Liberopoulos, G. 2018. "Performance Evaluation of a Production Line Operated Under an Echelon Buffer Policy." *IISE Transactions* 50 (3): 161–177.

Massim, Y., F. Yalaoui, L. Amodeo, E. Chatelet, and A. Zeblah. 2012. "Efficient Combined Immune-Decomposition Algorithm for Optimal Buffer Allocation in Production Lines for Throughput and Profit Maximization." *Computers & Operations Research* 37: 611–620.

Matondang, M. Z., and M. I. Jambak. 2010. "Soft Computing in Optimizing Assembly Lines Balancing." *Journal of Computer Science* 6 (2): 141–162.

Miller, B. L., and D. E. Goldberg. 1995. "Genetic Algorithms, Tournament Selection, and the Effects of Noise." *Complex Systems* 9 (3): 193–212.

Nahas, N., D. Ait-Kadi, and M. Nourelfath. 2006. "A New Approach for Buffer Allocation in Unreliable Production Lines." *International Journal of Production Economics* 103: 873–881.

Papadopoulos, C. T., J. Li, and Michael E. J. O'Kelly. 2019. "A Classification and Review of Timed Markov Models of Manufacturing Systems." *Computers & Industrial Engineering* 128: 219–244.

Pavai, G., and T. V. Geetha. 2019. "New Crossover Operators Using Dominance and Co-Dominance Principles for Faster Convergence of Genetic Algorithms." *Soft Computing* 23 (11): 3661–3686.

Robbins, H., and S. Monro. 1951. "A Stochastic Approximation Method." *Annals of Mathematical Statistics* 22: 400–407.

Schindl, D., and N. Zufferey. 2015. "A Learning Tabu Search for a Truck Allocation Problem with Linear and Nonlinear Cost Components." *Naval Research Logistics* 62 (1): 32–45.

Shaohui, Xi, Q. Chen, J. M. G. Smith, N. Mao, A. Yu, and H. Zhang. 2019. "A New Method for Solving Buffer Allocation Problem in Large Unbalanced Production Lines." *International Journal of Production Research*. doi:10.1080/00207543.2019.1685709.

Shi, L., and S. Men. 2003. "Optimal Buffer Allocation in Production Lines." *IIE Transactions* 35: 1–10.

Spinellis, D. D., and C. T. Papadopoulos. 2000. "Stochastic Algorithms for Buffer Allocation in Reliable Production Lines." *Mathematical Problems in Engineering* 5: 441–458.

Spinellis, D. D., C. T. Papadopoulos, and J. M. MacGregor. 2000. "Large Production Line Optimization Using Simulated Annealing." *International Journal of Production Research* 38: 509–541.

Suri, R. 1989. "Perturbation Analysis: The State of the Art and Research Issues Explained Via the GI/G/1 Queue." *Proceedings of the IEEE* 77 (1): 114–137.

Thevenin, S., and N. Zufferey. 2019. "Learning Variable Neighborhood Search for a Scheduling Problem with Time Windows and Rejections." *Discrete Applied Mathematics* 261: 344–353.

Thevenin, S., N. Zufferey, and J.-Y. Potvin. 2017. "Makespan Minimization for a Parallel Machine Scheduling Problem with Preemption and Job Incompatibility." *International Journal of Production Research* 55 (6): 1588–1606.

Thevenin, S., N. Zufferey, and M. Widmer. 2016. "Order Acceptance and Scheduling with Earliness and Tardiness Penalties." *Journal of Heuristics* 22 (6): 849–890.

Tomassini, M. 1999. "Parallel and Distributed Evolutionary Algorithms: A Review." *Evolutionary Algorithms in Engineering and Computer Science*, 113–133.

Turki, S., S. Hennequin, and N. Sauer. 2013. "Perturbation Analysis for Continuous and Discrete Flow Models: A Study of the Delivery Time Impact on the Optimal Buffer Level." *International Journal of Production Research* 51 (13): 4011–4044.

Vergara, H. A., and D. S. Kim. 2009. "A New Method for the Placement of Buffers in Serial Production Lines." *International Journal of Production Research* 47 (15): 4437–4456.

Weiss, S., J. A. Schwarz, and R. Stolletz. 2018. "The Buffer Allocation Problem in Production Lines: Formulations, Solution Methods, and Instances." *IISE Transactions* 5854 (2): 1–30.

Yegul, M. F., F. S. Erenay, S. Striepe, and M. Yavuz. 2017. "Improving Configuration of Complex Production Lines via Simulation-Based Optimization." *Computers & Industrial Engineering* 109: 295–312.

Zandieh, M., M. N. Joreir-Ahmadi, and A. Fadaei-Rafsanjani. 2017. "Buffer Allocation Problem and Preventive Maintenance Planning in non-Homogenous Unreliable Production Lines." *International Journal of Advanced Manufacturing* 91: 2581–2593.

Zhao, C., J. Li, N. Huang, and J. A. Horst. 2017. "Flexible Serial Lines with Setups: Analysis, Improvement, and Application." *IEEE Robotics and Automation Letters* 2 (1): 120–127.

Zufferey, N. 2012. "Metaheuristics: Some Principles for an Efficient Design." *Computer Technology and Applications* 3: 446–462.