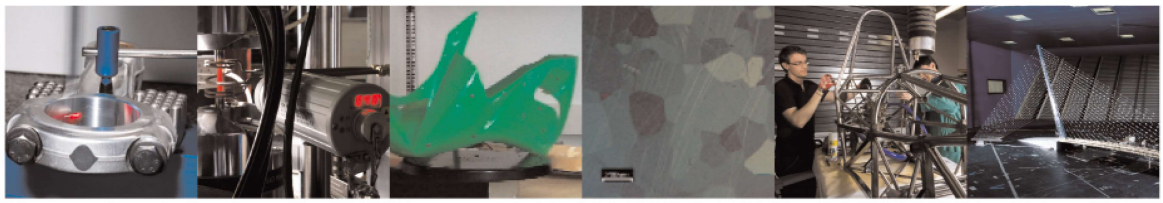

Multi-fidelity surrogate-based optimization for decomposed buffer allocation problems

Ziwei Lin, Nicla Frigerio, Andrea Matta, Shichang Du

This is a post-peer-review, pre-copyedit version of an article published in journal title. The final authenticated version is available online at: http://dx.doi.org/10.1007/s00291-020-00603-y

Noname manuscript No.
(will be inserted by the editor)

Multi-fidelity Surrogate-Based Optimization for Decomposed Buffer Allocation Problems

Ziwei LIN · Nicla FRIGERIO · Andrea MATTA · Shichang DU

Received: date / Accepted: date

Abstract The Buffer Allocation Problem (BAP) for flow lines has been extensively addressed in literature. In the framework of iterative approaches, algorithms alternate an evaluative method and a generative method. Since an accurate estimation of system performance typically requires high computational effort, an efficient generative method reducing the number of iterations is desirable, for searching for the optimal buffer configuration in a reasonable time. In this work, an iterative optimization algorithm is proposed which a highly accurate simulation is used as the evaluative method and a surrogate-based optimization is used as the generative method. The surrogate model of the system performance is built to select promising solutions so that an expensive simulation budget is avoided. The performance of the surrogate model is improved with the help of fast but rough estimators obtained with approximated analytical methods. The algorithm is embedded in a problem decomposition framework: several problem portions are solved hierarchically to reduce the solution space and to ease the search of the optimum solution. Further, the paper investigates a jumping strategy for practical application of the approach so that the algorithm response time is reduced. Numerical results

Z. Lin
Department of Industrial Engineering and Management, Shanghai Jiao Tong University, Shanghai, CHINA, Department of Mechanical Engineering, Politecnico di Milano, Milano, ITALY

N. Frigerio
Department of Mechanical Engineering, Politecnico di Milano, Milano, ITALY E-mail: nicla.frigerio@polimi.it

A. Matta
Department of Mechanical Engineering, Politecnico di Milano, Milano, ITALY

S. Du
Department of Industrial Engineering and Management, Shanghai Jiao Tong University, Shanghai, CHINA

are based on balanced and unbalanced flow lines composed of single-machine stations.

Keywords Buffer allocation problem · Multifidelity surrogate modeling · Simulation-optimization

1 Introduction

A production system can be seen as a set of resources interconnected by a material handling system where work-in-process might be held in buffers between two sequential stations. These buffers of parts help in reducing the propagation of blocking and starvation phenomena along the production system. However, dedicating space to maintain interoperative inventories is costly and extends the production lead time. For these reasons, the Buffer Allocation Problem (BAP) is an optimization problem of high importance for industries where there is a trade-off between productivity criteria and design and management costs.

The classical primal BAP considers the total allocated buffer capacity as the objective function and the throughput satisfaction as a constraint, this is known in literature as the primal problem [Gershwin and Schor, 2000]. The dual problem, also common in literature, maximizes the throughput under a constrained buffer capacity. This paper focuses on the primal problem. Furthermore, we address problems in which the processing times at servers follow general distributions and operational dependent failures might occur. With these assumptions, it is difficult to obtain accurate estimates of system throughput by using analytical methods. Therefore, simulation, despite being expensive in terms of execution, is frequently used as estimation method. Also, the solution space becomes wide as the number of stations increases and the search for the optimum gets harder. Hence, algorithms aim to obtain a good solution, with less simulation effort.

1.1 State of the Art for BAP

A recent and comprehensive review of BAP can be found in [Weiss et al., 2019] where a classification of state-of-the-art approaches is proposed. Solving methods are classified into three classes: explicit solutions, iterative optimization methods, and integrated optimization methods. The first class of *explicit solutions* provides a set of rules or established formulas describing the BAP. The methods in this class can only address BAP that are small in size, or with significant limitations due to the strong assumptions introduced to make the problem analytically tractable. The *integrated optimization methods* formulate the BAP into a mixed integer linear programming (MILP) model. For example, [Soyster et al., 1979] use an analytical representation of the problem. Other examples build a MILP to find a sample-exact solution, e.g. [Matta, 2008, Helber et al., 2011, Alfieri and Matta, 2012, Stolletz and Weiss, 2013].

Most of the references in literature follow *iterative optimization methods* to solve the BAP: a generative method selects promising buffer allocations and an evaluative method estimates the performance of the given candidate solution [Papadopoulos et al., 2009]. Markov chain analysis, decomposition methods [Gershwin, 1987], aggregation methods [Li and Meerkov, 2009], and simulation are used as the performance evaluation method with a clear trade-off between the accuracy and the computational effort. Enumeration, meta-heuristic and search-based algorithms are mostly used as the generative method. For instance, [Hillier, 2000] enumerates a set of the most promising solutions, [Matta et al., 2012] uses a surrogate-based optimization algorithm with Kriging (more details on Kriging as in [Sacks et al., 1989]), [Kose and Kilincci, 2015] combines simulated annealing and genetic algorithm for exploring and exploiting the search spaces, [Shi and Gershwin, 2016] guides the search with the gradient calculated analytically in the evaluative method. Nested partition and branch-and-bound are also used [Shi and Men, 2003, Dolgui et al., 2007]. An efficient generative method can reduce the number of algorithm iterations before reaching a near-optimal solution and can save the effort required in the evaluative method.

Iterative optimization methods are frequently applied in real cases. Unlike integrated optimization methods, iterative optimization methods treat the evaluation method as a black-box, i.e., the inner structure of the evaluation method is not considered in the optimization algorithm, which makes them easy to implement. Nevertheless, the lack of knowledge about the throughput function (e.g. gradient information), together with the large search space and the time-consuming evaluation method affect the efficiency of commonly used searching methods (e.g. enumeration method, meta-heuristic, gradient-based method).

Some state-of-the-art approaches uses problem decomposition to reduce the computational effort. The BAP is divided into several sub-problems that are easier to solve than the final problem (i.e., the non-decomposed problem). The solution of each sub-problem helps to solve the final problem. For example, [Shi and Gershwin, 2016] decomposes the system into several sub-systems, each representing an overlapping portion of the system. The BAP is solved for each sub-system independently. Then, the near-optimal buffer allocation of the system is found by combining the sub-system's solutions. Another example is found in [Weiss and Stolletz, 2015] and [Weiss et al., 2018]. The authors decompose the system into several sub-systems whose dimension provides a hierarchical ordering. Starting from the lower hierarchy (i.e., single-dimensional BAP), local solutions are found and create exact bounds for higher hierarchies that are solved afterwards. Despite the advantages of problem decomposition approaches, the knowledge obtained at a certain hierarchy is exploited only in the form of bounds. A large amount of data about the sub-system's performance is wasted when the algorithm moves to higher hierarchies, although these sets of data might contain information that could increase the search efficiency.

1.2 Contribution

This paper proposes an iterative optimization algorithm for the primal BAP in which a surrogate-based optimization method is used as the generative method to save the effort in the evaluative method, which is simulation. The algorithm is embedded in a problem decomposition framework to save the search effort in the generative method.

Simulation is used as the evaluative method to accurately estimate the system's throughput, but it is time-consuming. A surrogate model can be created from few simulated data to predict the system throughput of the buffer configurations that have not been simulated. Thus, promising solutions can be pointed out quickly by the surrogate model and the budget for the evaluative method can be carefully allocated. Throughout the paper, this budget is referred to as the *simulation budget* (i.e., the number of candidate solutions that are evaluated using simulation). The Extended Kernel Regression (EKR [Lin et al., 2019]) method is used in this paper to create the surrogate model since it can improve the accuracy of the build surrogate model by combing the simulation data with rough but fast estimators, e.g. analytical methods and coarse simulations. The surrogate model might be biased in some areas of the domain, which may lead to a wrong promising solution. Therefore, both the predicted system performance and the quality of the built surrogate model are considered to select the promising solutions.

The proposed algorithm is embedded in a problem decomposition framework [Weiss and Stolletz, 2015], in which the original problem is divided into sub-problems with different hierarchies. The optimal solutions of sub-problems in lower hierarchies provide lower bounds to sub-problems in higher hierarchies according to the features of the system. Therefore, the search space in the generative method can be reduced and the search effort can be saved. In addition, the estimates obtained during solving a certain sub-problem can be re-used throughout the problem decomposition hierarchy. Despite these re-used estimates being approximated, they represent a part of the system and can improve the accuracy of the surrogate model.

A preliminary version of the algorithm has been analyzed in recent literature [Frigerio et al., 2018]. The work is herewith extended by considering the prediction error of the surrogate model and by including an analytical method in the creation of surrogate models. A surrogate-based optimization method is proposed for BAP in [Matta et al., 2012], in which a surrogate model guides the search in the generative method. Differently from [Matta et al., 2012], where the Kriging technique is used to create the surrogate model with data from a single source, a multi-fidelity surrogate model is created in this paper. The use of multiple sources can increase the prediction performance of the built surrogate model, thereby improving the quality of selected promising solutions. Also, [Matta et al., 2012] considers only the system performance estimates and does not include the prediction error, i.e., the quality of the surrogate model.

A set of numerical cases shows the accuracy of the surrogate model in terms of prediction error. These cases also show that the proposed iterative algorithm is efficient and the benefit of the involvement of an analytical method in the construction of the surrogate model is significant. The proposed algorithm is more effective when the total buffer capacity of the optimal buffer configuration is high, i.e. when the required throughput is high. Considering the decomposed problem, reusing data can also improve the efficiency of the algorithm. When problem dimension becomes high (i.e., long lines), the trade-off between the effort to solve sub-problems and the size of cut search space has an important impact on the computational time. Some strategies are investigated to improve the results in these cases.

1.3 Paper Outline

The paper is divided into five sections. After introducing the problem and the related literature in Section 1, the proposed iterative algorithm with a surrogate-based generative method is described in Section 2. Section 3 describes how the proposed algorithm is embedded in a problem decomposition framework. Numerical results are provided in Section 4. Section 5 concludes the paper.

2 A Surrogate-Based Solving Algorithm

In this section, we formulate the problem in question, and we provide the description of the proposed algorithm.

2.1 Problem Description and Modeling

The system being studied is a classical flow line composed of S single-server stations and $S-1$ finite intermediate buffers. For the sake of simplicity, the first machine is assumed to be never starved of raw parts and the last machine is never blocked (i.e., saturated supply and saturated demand). The blocking after service rule is used for stations, although the problem is similar for the blocking before service rule. Let us use x_s to denote the buffer capacity allocated to the buffer behind station s and $\mathbf{x} = \{x_1, x_2, \dots x_{S-1}\}$ to denote the vector of decision variables describing the buffer allocation along the line.

The total buffer capacity of the line is defined as follows:

$$z(\mathbf{x}) = \sum_{s=1}^{S-1} x_s. \tag{1}$$

The buffer capacity needs to be allocated in order to minimize the total buffer capacity $z(\mathbf{x})$ while a certain throughput target y_{target} is reached. We assume

that the capacity x_s of buffer s is limited by the user-defined upper bound B_s. The BAP is formulated as follows:

$$\min \{z(\mathbf{x}) \mid y(\mathbf{x}) \geq y_{\text{target}} \ ; \ 0 \leq x_s \leq B_s, x_s \in \mathbb{N}, \forall s = 1, \ldots, S-1\} \quad (2)$$

where the expected throughput $y(\cdot)$ of the system is a non-linear function of decision variables $\mathbf{x}$. We model processing times T_s with $s = 1, \ldots, S$ as generally distributed random variables. Transportation times are negligible or already included in the processing times. Operational dependent failures are also included in the processing time distributions.

2.2 Algorithm Main Structure

The main structure of the proposed algorithm is represented in Figure 1. The algorithm belongs to the category of iterative approaches and it alternates two main parts: evaluation and generation [Papadopoulos et al., 2009]. In a general iteration i, buffer configuration $\mathbf{x}_i$ is identified as promising by the *generative method*. Therefore, system performance $y(\mathbf{x}_i)$ can be accurately obtained using a simulation model as the *evaluative method*.

We assume the line processes W parts, where W_0 parts correspond to the warm-up phase. Given proper values for W and W_0, the simulation model provides an accurate estimate $y(\mathbf{x}_i)$ of the expected throughput obtained with buffer allocation $\mathbf{x}_i$.

At the first iteration, the surrogate model is built, as described in Section 2.3, starting from an initial set $\mathbb{X}^0$ of n_0 candidate solutions. The initial design $\mathbb{X}^0$ is evaluated using simulation and the generative method can start. Then, the *generative method* solves an optimization problem as described in Section 2.4. Within this phase, a surrogate model is built to provide both the estimate of the expected throughput $\hat{y}(\cdot)$ and the estimated square prediction error $\hat{s}^2(\cdot)$. Hence, the promising solution $\mathbf{x}_i$ is found and evaluated using simulation. In subsequent iterations, the surrogate model is updated with new observed (or simulated) data. When the stopping condition is satisfied, the algorithm stops.

2.3 A Multi-fidelity Surrogate Model for BAP

Assume that a certain number of models is available to provide the system performance estimates. In particular: one time-consuming *High-Fidelity* (HF) model, i.e., the simulation model, providing highly accurate estimates $y(\mathbf{x})$, and a certain number of *Low-Fidelity* (LF) models, i.e., analytical methods, coarse simulations, and meta-models, providing approximated estimates quickly.

We adopt the Dallery-David-Xie (DDX) algorithm [Dallery et al., 1988] from the literature to provide the LF estimate $y_{\text{DDX}}(\mathbf{x})$ of system performance. The reason for this choice is its easiness of implementation without critical

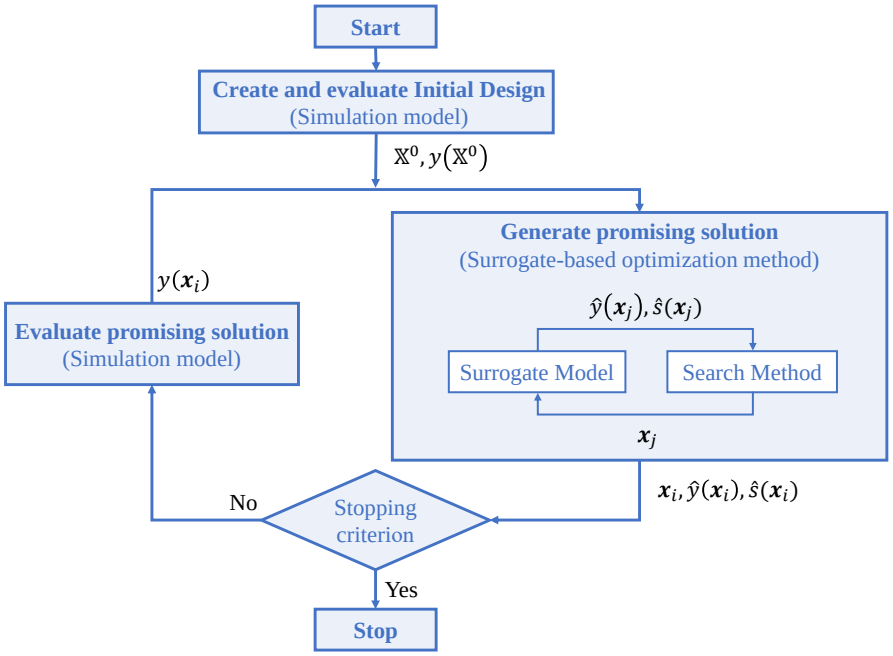

Fig. 1 Structure of the solving algorithm.

numerical issues. Other algorithms could be adopted without changing the approach, e.g. [Tolio and Matta, 1998, Liberopoulos et al., 2006, Li and Meerkov, 2009, Colledani and Gershwin, 2013]. Also, more than one method can be included without requiring a large extension to the developed algorithm.

The initial design $\mathbb{X}^0$ is composed of n_0 *design points* sampled using a space filling design (we used, as an example, a Latin Hypercube Sampling (LHS) [McKay et al., 1979]). The system performance at the design points is calculated using both the HF and LF models.

The creation of the surrogate model is performed using the Extended Kernel Regression (EKR) method [Lin et al., 2019], based on the available information, i.e, both HF and LF estimates of design point in set $\mathbb{X}^0$, and the LF estimate of the unknown point $\mathbf{x}$:

$$y_{\mathrm{EKR}}(\mathbf{x}|y(\boldsymbol{u}), y_{\mathrm{DDX}}(\boldsymbol{u}), \forall \boldsymbol{u} \in \mathbb{X}^0, y_{\mathrm{DDX}}(\mathbf{x})) \Rightarrow \hat{y}_{\mathrm{EKR}}(\mathbf{x}), \hat{s}^2_{\mathrm{EKR}}(\mathbf{x}). \quad (3)$$

The system performance estimate $\hat{y}_{\mathrm{EKR}}(\mathbf{x})$ at a certain buffer allocation $\mathbf{x}$ is obtained as well as its estimated square error $\hat{s}^2_{\mathrm{EKR}}(\mathbf{x})$ that indicates the prediction error of the surrogate model. More details on how to build the surrogate model using EKR are provided in Appendix **??**.

2.4 Optimization Procedure

In the generative method, the built surrogate model is used to guide the search for a promising solution. The surrogate model could be biased and the promising solution provided could be incorrect. Therefore, to balance the exploitation (to find the best solution according to the surrogate model) and the exploration (to improve the quality of the surrogate model), the Expected Improvement (EI) criterion ([Mockus et al., 1978, Jones et al., 1998]) is applied. It assumes that the true value of the system's performance follows a prior distribution, e.g. normal distribution, in which the mean is affected by the system performance estimate provided and the variance is affected by the prediction error. Then, the solution that has the maximal EI compared to the current best solution is considered as the most promising solution. The EI of an unknown solution $\mathbf{x}$ is herewith defined as follows:

$$EI(\mathbf{x}) = \left(z(\mathbf{x}^{\text{best}}) - z(\mathbf{x})\right) \cdot P\left(y(\mathbf{x}) \geq y_{\text{target}}\right). \tag{4}$$

where the current best solution $\mathbf{x}^{\text{best}}$ is defined as the configuration that has been simulated, satisfies the throughput target y_{target}, and has the lowest total buffer capacity. The first term in expression (4) is the distance of the $\mathbf{x}$ configuration from the current best $\mathbf{x}^{\text{best}}$ in terms of total buffer capacity. The second term is the probability that configuration $\mathbf{x}$ satisfies the throughput target. When EI is high, it is more likely that the current best can be improved.

The probability that a solution is feasible $P\left(y(\mathbf{x}) \geq y_{\text{target}}\right)$ is calculated using the system performance estimate $\hat{y}_{\text{EKR}}(\cdot)$ and the estimated square error $\hat{s}^2_{\text{EKR}}(\cdot)$ provided by the surrogate model:

$$P\left(y(\mathbf{x}) \geq y_{\text{target}}\right) \approx \Phi\left(\frac{\hat{y}_{\text{EKR}}(\mathbf{x}) - y_{\text{target}}}{\hat{s}_{\text{EKR}}(\mathbf{x})}\right). \tag{5}$$

The normal distribution is used according to [Lin et al., 2019] which is obtained by applying the Central Limit Theorem.

At each iteration i, the following optimization problem is solved to obtain the promising point $\mathbf{x}_i$:

$$\mathbf{x}_i = \arg\max_{\mathbf{x}} EI(\mathbf{x}) \tag{6}$$

$$s.t.: \quad z(\mathbf{x}) < z(\mathbf{x}^{\text{best}}) \tag{7}$$

$$x_s \leq B_s, x_s \in \mathbb{N}^+, \forall s. \tag{8}$$

The above optimization problem is bounded by the current best solution using constraint (7), this is to avoid wasting effort in unpromising areas. To solve this problem, algorithms such as meta-heuristics, random searches, etc. can be used to provide good solutions quickly. In Section 4, a common Genetic Algorithm from the Matlab package will be used for experiments, but other algorithms could be successfully adopted.

2.5 Stopping Condition

The EI measure defined in Section 2.4 is an indicator of solution quality and it can be used to interrupt the algorithm. As $EI(\mathbf{x}_i)$ decreases, the solution approaches the optimum and it is difficult (rather than not possible) to find an improvement. Therefore, the algorithm is stopped when the maximal EI, found in iteration i, is below a certain threshold EI_{target}:

$$EI\left(\mathbf{x}_i\right) \leq EI_{\text{target}} \tag{9}$$

Under condition (9), the algorithm returns the current best solution $\mathbf{x}^{\text{best}}$. Otherwise, the algorithm performs a new iteration following a sequence of steps:

- The most promising point $\mathbf{x}_i$ is evaluated using the HF model.
- If $y(\mathbf{x}_i) \geq y_{\text{target}}$, the current best solution is updated, i.e., $\mathbf{x}^{\text{best}} = \mathbf{x}_i$.
- The surrogate model is updated by adding $\mathbf{x}_i$ to the set of design points $\mathbb{X}^0$.
- The iteration number is updated: $i = i + 1$.

It is noteworthy that the $\mathbf{x}^{\text{best}}$ is selected as the upper bounds of the solution at the beginning of the algorithm.

The algorithm accuracy can be tuned by decreasing target EI_{target}. However, as EI_{target} decreases the computational time also increases because the stopping condition becomes harder to satisfy.

3 The Algorithm Applied in the Problem Decomposition Framework

The algorithm proposed in Section 2 is embedded in a problem decomposition framework where the main BAP is decomposed into several sub-problems of smaller dimension. Hierarchical problem decomposition methods are widely used in optimization when the scale of the problem is large. Thus, a bottom-up approach is followed as described in Section 3.1. Also, when the BAP is decomposed, models with different detail levels are used to represent each sub-problem, and bounds are created as in Section 3.2. An innovative approach is proposed to exploit the knowledge, in addition to the bounds, derived from a certain sub-problem (Section 3.3).

3.1 Problem Decomposition Approach

Adopting the problem decomposition approach of [Weiss and Stolletz, 2015], the system is divided into several sub-systems assuming that the first station of each sub-system has an unlimited supply (i.e., saturated supply) and that the last station is never blocked (i.e., saturated demand). Each sub-system, denoted as $M(\ell, j)$, represents a portion of $\ell + 1$ sequential stations of the

system. Index j indicates the first machine in the sub-system and $j+\ell$ indicates the last. A total of $S-1$ hierarchies is created and each hierarchy $\ell \in [1, S-1]$ includes $S-\ell$ sub-systems, i.e., $M(\ell, j)|j = 1, \ldots, S-\ell$. Figure 2 represents an example of system decomposition.

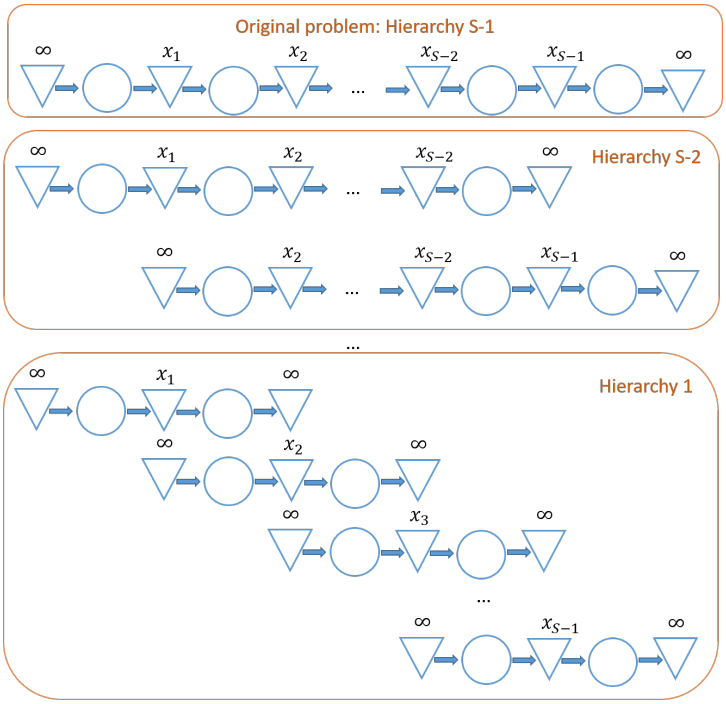

Fig. 2 System decomposition in sub-systems with stations (represented with circles) and buffers (represented with triangles). As an example, three hierarchies are reported: the final and complete system of hierarchy $S-1$, hierarchy $S-2$ composed of two sub-systems of $S-2$ sequential machines, and the lowest hierarchy composed of $S-1$ sub-systems of two sequential machines.

Each sub-system $M(\ell, j)$ implies a BAP whose dimension ℓ is smaller compared to that of the complete system, i.e., $S-1$. Let us denote the optimal solution of sub-system $M(\ell, j)$ as the (ℓ)-tuple of buffer capacities from x_j to $x_{j+\ell-1}$:

$$\mathbf{x}^{\text{best}}_{(\ell,j)} = \{x^{\text{best}}_j, \ldots, x^{\text{best}}_{j+\ell-1}\}. \tag{10}$$

Figure 3 represents the main framework of the proposed algorithm embedded in the bottom-up decomposition approach. The overall algorithm consists of the following steps:

i) Following a bottom-up approach, the first level of hierarchy $\ell = 1$ is addressed starting from machine $j = 1$.

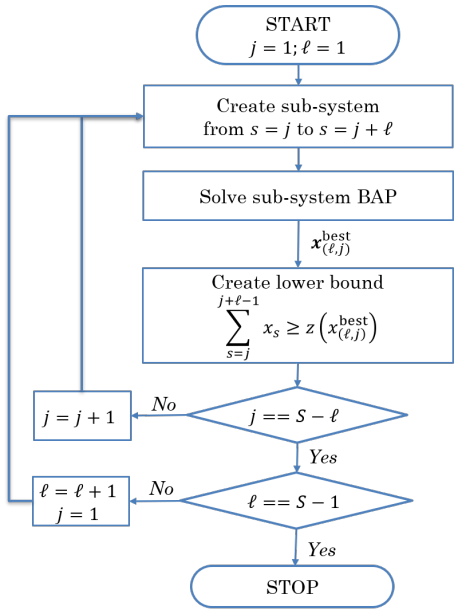

Fig. 3 The algorithm embedded in the problem decomposition framework.

ii) Focusing on sub-system $M(\ell, j)$, the solution $\mathbf{x}^{\text{best}}_{(\ell,j)}$ of the associated BAP is found using the algorithm described in Section 2.
iii) A lower bound is created as in Section 3.2.
iv) Steps (ii), (iii), and (iv) are repeated with the next sub-system ($j = j+1$) or with the next hierarchy level ($\ell = \ell + 1; j = 1$) until the final problem $M(S-1, 1)$ is solved.

Given ℓ and j, the sub-problem in question belongs to hierarchy level ℓ, it includes stations $s = j$ to $s = j+\ell$, and it has ℓ dimensions. Note that all sub-systems in the lower hierarchy have been solved previously since a bottom-up approach is used, and their solutions $\mathbf{x}^{\text{best}}_{(L,J)} | L < \ell, J \in [1, S-\ell]$ exist.

3.2 Creation of Lower Bounds

The isolated throughput of sub-system $M(\ell, j)$ is higher than that of a larger system with the same corresponding buffer allocation [Weiss and Stolletz, 2015]. For example, assume that the best buffer allocation for system $M(2, 1)$ is $\mathbf{x}^{\text{best}}_{(2,1)} = \{3, 5\}$ for a certain throughput target. This implies that sub-systems $M(L, 1) | L > 2$, which include $M(2, 1)$, require a total buffer capacity among the first two buffers of at least 8 buffer slots, i.e., $x_1 + x_2 \geq z(\mathbf{x}^{\text{best}}_{(2,1)}) = 8$.

The solution $\mathbf{x}^{\text{best}}_{(\ell,j)}$ of sub-problem $M(\ell, j)$ provides a lower bound to all sub-problems belonging to a higher hierarchy ($L > \ell$) and including sub-system

$M(\ell, j)$. Therefore, bounds are formalized as follows:

$$z(\mathbf{x}) \geq z(\mathbf{x}^{\text{best}}_{(\ell,j)}) \tag{11}$$

and they are effective for all sub-systems belonging to the following set:

$$M(L, J)|L > \ell; J \in [\max(1, j + \ell - L), j]. \tag{12}$$

These lower bounds, introduced by [Weiss and Stolletz, 2015], are added into the optimization problem (cf. equations (6)–(8)). They can narrow the search space and accelerate the generative method.

3.3 The Re-use of Data

Let us consider sub-system $M(\ell, j)$ and its associated BAP. According to the bottom-up solving approach, sub-systems of lower hierarchies, i.e., $L < \ell$, have already been solved. As a consequence, the surrogate models of these sub-systems are available and can provide coarse estimates of sub-system $M(\ell, j)$. These coarse estimators can be reused as low-fidelity models to improve the prediction performance of the surrogate model in sub-system $M(\ell, j)$. The algorithm can use the following models for creating the surrogate model in sub-system $M(\ell, j)$:

- The HF simulation model of sub-system $M(\ell, j)$: $y^{(\ell,j)}$.
- The LF model of sub-system $M(\ell, j)$ created with DDX method: $y^{(\ell,j)}_{\text{DDX}}$.
- The LF surrogate models of sub-systems $M(\ell - 1, j)$ and $M(\ell - 1, j + 1)$: $y^{(\ell-1,j)}_{\text{EKR}}$ and $y^{(\ell-1,j+1)}_{\text{EKR}}$.

The re-use of data might be helpful and, in general, many LF models can be included. It is noteworthy that the computational time of the EKR model increases with the number of LF models included and the surrogate model becomes redundant as the sub-system hierarchy increases. The EKR method autonomously identifies which models are more helpful in different regions of the domain and assigns area-based weights accordingly. Therefore, as an additional improvement, the algorithm removes the LF models that have low weights in the whole domain. This feature extends the original EKR method in [Lin et al., 2019] to further save computational time.

4 Numerical Results

In this section, experiments are carried out and reported to show the efficiency and effectiveness of the proposed method. Section 4.1 describes the scenarios used to obtain numerical results. Section 4.2 analyzes the predictive performance of the surrogate model. Then, numerical results are divided into two main parts: the first part (Section 4.3) focuses on the performance of the algorithm when applied directly to the final problem ($M(S - 1, 1)$), and the second part (Section 4.4) is devoted to the algorithm within the decomposition framework.

4.1 Scenario Description

Numerical results are based on both balanced (denoted as BAL) and unbalanced production lines with a maximum buffer capacity of $B_s = 30$. Scenarios are created by varying the position of the bottleneck: MID denotes a line with a bottleneck in the middle and B2 denotes a line with two bottleneck machines. Short lines are analyzed firstly, and the analysis on long lines follows. Furthermore, two production rate targets are used to represent high-target (large allocated buffer capacity required) and low-target (small allocated buffer capacity required) situations. The analyzed scenarios are summarized in Table 1. Notation Mz-XXX-X is used: Mz indicates a line of z machines, XXX indicates the position of the bottleneck, and the last letter indicates the throughput target (H = high and L = low).

The processing times are assumed deterministic: 0.5 minutes are required to process a part in balanced lines, whilst 0.45 minutes are required in unbalanced lines where only the bottleneck machines require 0.5 minutes. Further, machines are unreliable and the Times To Repair (TTR) follow a Weibull distribution with $\lambda = 5.64$ and shape $k = 2$. Times to Failure (TTF) are correlated to TTR so that TTF = TTR + Z where Z is a random variable distributed accordingly to a Weibull distribution with scale $\lambda = 22.15$ and shape $k = 1.5$. The correlation between TTR and TTF is used to model that failures requiring long repair times occur less frequently.

Scenario	Number of machines	Bottleneck position	y_{target}[ppm]
M5-BAL-H	5	none	1.52
M5-BAL-L	5	none	1.44
M5-MID-H	5	middle, i.e. $s = 3$	1.60
M5-MID-L	5	middle, i.e. $s = 3$	1.51
M5-B2-H	5	two, i.e. $s = 2$ and $s = 4$	1.60
M5-B2-L	5	two, i.e. $s = 2$ and $s = 4$	1.51
M15-BAL-H	15	none	1.60
M15-BAL-L	15	none	1.44
M15-MID-H	15	two, i.e. $s = 5$ and $s = 11$	1.60

Table 1 Scenario description with production rate target expressed in parts per minute [ppm].

The algorithm and the methods included are implemented in the Matlab environment. The Welch method [Law and Kelton, 2000] is used to identify simulation initial transitory which is $5 \cdot 10^4$ parts for 5 machine lines and $3 \cdot 10^5$ for long lines. Simulation length is $2.5 \cdot 10^5$ parts for short lines and $5 \cdot 10^5$ parts for long lines.

The DDX method requires failures and repairs follow geometric distributions with rates p and r respectively. These parameters have been estimated from the TTF and TTR distributions, i.e., $\hat{p} = 0.02$ and $\hat{r} = 0.1$, to properly represent each machine. Furthermore, DDX cannot consider the correlation between TTR and TTF.

4.2 Surrogate Model Prediction Performance

The Mean Absolute Percentage Error ($MAPE$) of the estimated production rate is considered as an accuracy index. The $MAPE$ is defined as follows:

$$MAPE = \frac{1}{n_c} \sum_{j=1}^{n_c} \frac{|y(\mathbf{x}_j) - \hat{y}(\mathbf{x}_j)|}{y(\mathbf{x}_j)} \cdot 100[\%] \tag{13}$$

where $\hat{y}$ is the estimator of the production rate to be compared and $\mathbf{x}_j$ is a *checkpoint*, i.e., a buffer allocation sampled from the solution space to assess the prediction performance of the estimator. Independently from design points, n_c checkpoints are sampled using LHS. Using HF simulation estimate y as a reference, we compare the error of the EKR method and that of the standard Kernel Regression (KR) method [Wand and Jones, 1995] which does not use LF data in surrogate creation. The DDX error is also included in the comparison. In this analysis, the optimization is not performed, because the scope is to focus only on the accuracy of the surrogate model.

We only provide results obtained on balanced lines of 5 and 15 equal machines, because their performance is generally more difficult to predict than unbalanced lines. Similar insights can be obtained with unbalanced lines. In these experiments, the simulated sample path changes at each design point and at each checkpoint. Using $n_c = 10^4$ checkpoints, the DDX method obtains: $MAPE = 8.18\%$ for M5-BAL, and $MAPE = 8.21\%$ for M15-BAL. Figure 4 represents the $MAPE$ obtained with EKR and KR methods as the number of design points n_0 increases. The EKR method is more accurate than the KR method, meaning that the use of DDX estimates provides useful information. From another perspective, the accuracy of DDX is highly improved by using few simulation data.

4.3 Benefit of Surrogate-Based Optimization

The proposed algorithm is denoted as "KR" and "EKR" respectively when the surrogate model is created with the KR and EKR methods. The GA available in the Matlab package is used for selecting candidate solutions $\mathbf{x}_i$ by maximizing the expected improvement $EI(\mathbf{x})$ in EKR and KR algorithm settings. Hence, KR and EKR stops when $EI_{\text{target}} = 0$ is reached. The reasons for choosing GA are its performance in combinatorial problems and the availability of a code in the Matlab package. As already explained in Section 2.4, other algorithms could be adopted or developed.

Results have been compared with those obtained by an iterative algorithm, labeled "SIM", that does not use a surrogate-based method but uses a pure GA as the generative method. Therefore, SIM stops when GA stops. ($EI_{\text{target}} = 0$ is used for M5-scenarios, $EI_{\text{target}} = 0.02$ for M15-scenarios). The DDX method, in the evaluated scenarios, underestimates the system throughput. In most of the evaluated scenarios, the DDX output for the combination

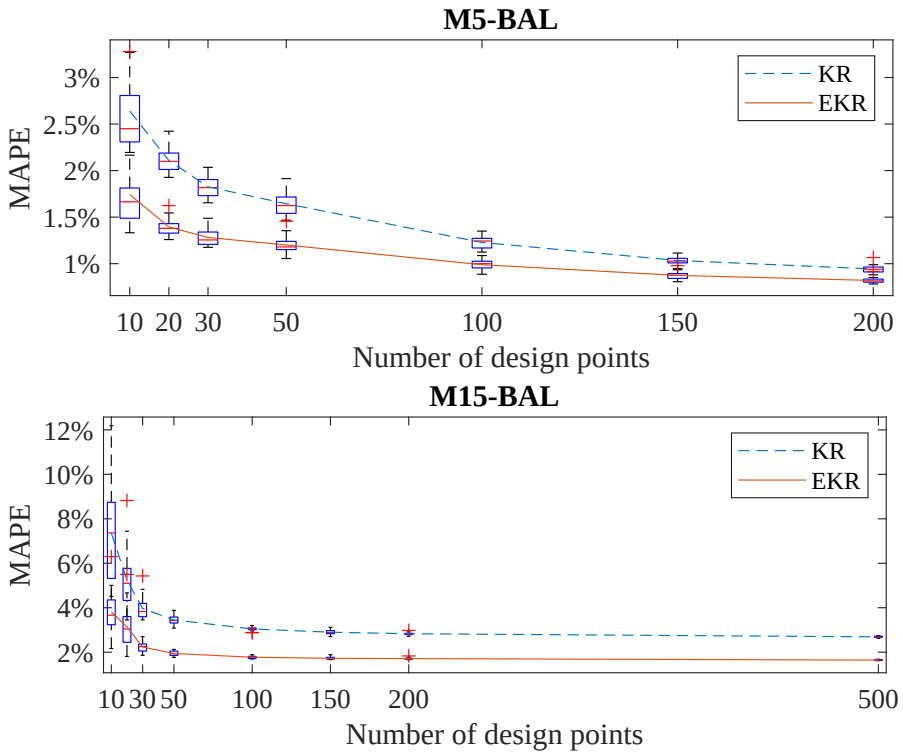

Fig. 4 $MAPE$ of the surrogate models, created with the EKR and KR methods, for scenarios M5-BAL and M15-BAL as the number of design points n_0 increases. Boxplots are created with 20 algorithm replications by varying the initial design $\mathbb{X}^0$.

of the buffer capacity upper bounds is lower than the defined throughput target, i.e., all solutions are infeasible according to the DDX outputs. Therefore, the DDX method cannot provide a promising solution to be simulated. For this reason, results obtained using DDX instead of the surrogate model are not included in the comparison. For this reason, results obtained using DDX instead of the surrogate model are not included in the comparison.

The parameters used for the GA embedded in SIM, KR and EKR have been calibrated. The GA selects the best candidates using a fitness scaling function based on candidate ranking. At each iteration, new candidates are generated (population size PS): a certain elite is guaranteed to survive (elite fraction EF) and new candidates are generated with a crossover function (crossover fraction CF) or with a mutation function. We used a scattered crossover function (a random binary vector identifies the variables from parents) and a gaussian mutation function (unitary scale and shrink factor). The algorithm stops when the maximum number of generations MG is reached or when the average relative change in the fitness function value over a certain number of stall generations SG is less than a certain tolerance T. Factors PS, EF, CF and T

have been selected as in Table 2. Factors SG and MG are tuned to stop the GA more efficiently.

Table 2 Selected parameters for GA.

Parameter	Value
Population Size PS	50
Elite Fraction EF	0.05
Crossover Fraction CF	0.8
Tolerance T	1e-6
Stall Generations SG	20 for M5; 8 for M15
Max Generations MG	1000

KR starts with the initial budget $n_0 = 32$ simulations, dedicated to evaluating design points, and performs a single simulation run at each iteration. Similarly, EKR starts with $n_o = 12$. Whereas, SIM performs 50 simulations at each iteration (therefore it starts at $n_0 = 50$). The current best solution improves as simulation budget n increases and tends to the optimum.

In order to reduce solution variability, the simulated sample-path used to evaluate configuration $\mathbf{x}$ is fixed for each scenario. Therefore, algorithm replicates differ in terms of creation of the surrogate model and in the search performed by the generative method.

4.3.1 Analysis of Algorithm Performance

Figure 5 shows the comparison for scenario M5-BAL-H. Of the KR and EKR algorithms, EKR obtains the best performance on average. Furthermore, compared to SIM, both KR and EKR converge quickly to the optimum which is obtained after around 1000 simulations by the SIM algorithm. A similar conclusion can be drawn with other scenarios as collected in Table 3, which shows the mean number of simulations required to reach the optimum for different algorithms. The objective value of the optimal solution, i.e., $z(\mathbf{x}^*)$, of each scenario is reported in Table 3 and has been validated using the algorithm proposed in [Weiss and Stolletz, 2015].

Scenario	$z(\mathbf{x}^*)$	Simulation Budget n		
		SIM	KR	EKR
M5-BAL-H	63	1060 ± 170	196 ± 31	78 ± 14
M5-BAL-L	39	592 ± 72	159 ± 25	35 ± 8
M5-MID-H	55	1693 ± 326	289 ± 48	46 ± 7
M5-MID-L	35	786 ± 107	214 ± 39	95 ± 22
M5-B2-H	83	675 ± 79	217 ± 34	122 ± 21
M5-B2-L	45	958 ± 116	188 ± 28	39 ± 6

Table 3 The simulation budget n required to obtain the optimum $z(\mathbf{x}^*)$ is reported according to the algorithm used (mean and corresponding 95% confidence interval is computed over 50 algorithm replications).

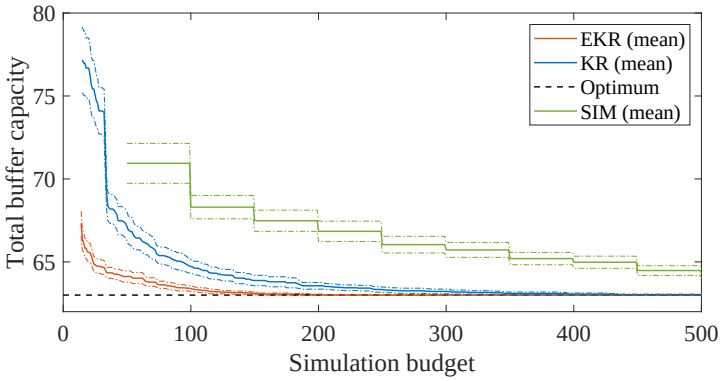

Fig. 5 Comparison of algorithm performance for scenario M5-BAL-H. The mean of 50 algorithm replications and its 95% confidence interval (dotted lines) is represented. Lines start when all replications reach a feasible solution. When the line of the mean reaches the optimum (i.e., 63 for this case), it means that all replications obtain the optimum.

Moreover, it is noteworthy that the accuracy of the solution obtained using the proposed algorithm varies according to the stopping condition used, i.e., the EI_{target}. The double effect on solution accuracy and simulation budget n can be analyzed as in Figure 6. For both algorithms (KR and EKR), the number of simulations used before the algorithm stops decreases as the EI_{target} increases. With $EI_{\text{target}} = 0.2$ the algorithms stop at first iteration so that the simulation budget n is equal to the initial budget n_0 (the initial budgets for KR and EKR are set to be the same, i.e., $n_0 = 16$, in this experiment for comparison purposes). On the contrary, the higher the EI_{target}, the lower the solution quality because the value of the total buffer capacity increases. In the figures, the average total buffer capacity with $EI_{\text{target}} = 0.2$ is lower than that with $EI_{\text{target}} = 0.15$ because of the sampling noise. According to the 95% confidence interval, the results of these two EI_{target} values are not statistically different. With the same value for EI_{target}, the EKR performs better than KR.

Similar conclusions can be drawn for long lines, i.e., scenarios M15-BAL-L and M15-MID-H. Figure 7 represents how the solution improves as simulation budget n increases. The *best found* solution among all algorithm replications is 189 for M15-BAL-L and 226 for M15-MID-H. Despite not knowing the optimum, we assume that the *best found* is the near-optimum solution of reference.

SIM and KR perform similarly, whereas EKR obtains good solutions after few simulations. The surrogate model created by KR is built with few initial pieces of data ($n_0 = 56$ for these scenarios) and, given the high-dimension of the problem, it does not perform well in estimating system performance. Therefore, the quality of the promising points, provided by the KR-based generative method, is low and the algorithm is slow in improving the objective function. On the contrary, despite the initial budget being low, the prediction accuracy of the surrogate model built with the EKR is highly improved by the

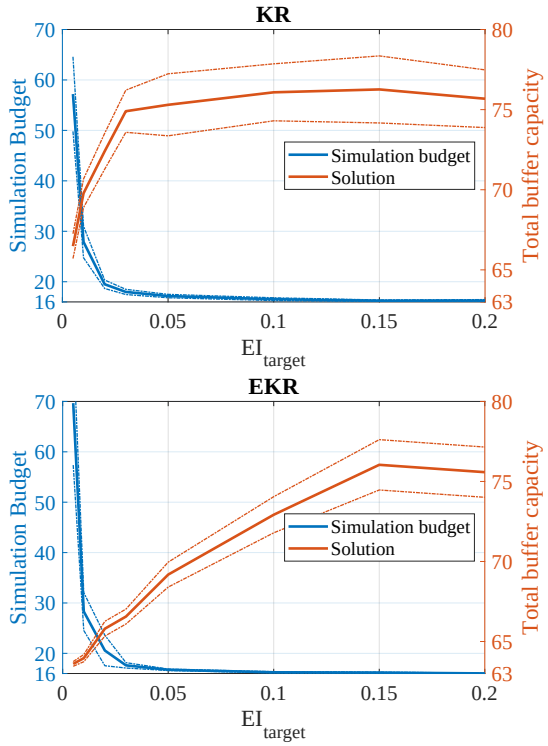

Fig. 6 Final solution obtained and number of simulations used (mean of 50 algorithm replications and 95% confidence interval) for M5-BAL-H varying EI_{target}.

involvement of the DDX method. A good and feasible solution is found after a few runs of the generative method (the cliff down in Figure 7).

4.3.2 A Note on Computational Time

The proposed algorithm is efficient in obtaining a good solution within few iterations, i.e., with limited simulation budget. Although the execution of the surrogate model is fast, at each iteration the generative method executes the surrogate model many times, thus it might lead to a high computational time. Nevertheless, the computational time required by the generative method is not affected by the running time of the simulation model, which is involved only in the evaluation phase. As a consequence, the proposed approach maintains efficiency in problems in which the evaluation method is highly time-consuming as well.

For the evaluated cases, the simulation of short (long) lines requires on the average 0.09 seconds (0.41 seconds) with Matlab2018b on a laptop Intel(R) Core(TM) i7-6600U with 2.6GHz and 16GB of RAM. The total simulation time accounts of around 10% of the total time required. The rest of the time includes the creation and update of the surrogate model and the generative

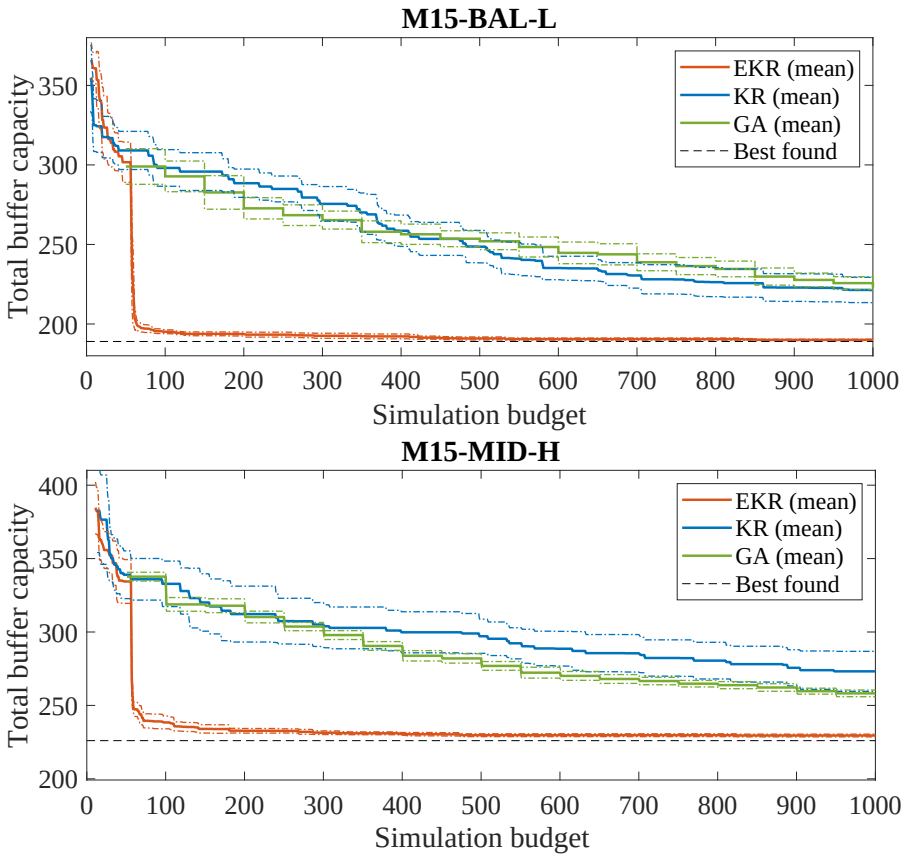

Fig. 7 Comparison of algorithm performance for scenarios M15-BAL-L and M15-MID-H. The mean of 10 algorithm replications and its 95% confidence interval (dotted lines) is represented for SIM, EKR, and KR.

method, i.e., the execution of the DDX method and the surrogate model to provide estimates. This time can be further reduced by improving the optimization technique.

It takes on average about 65 seconds to reach the optimum in scenario M5-BAL-H, and about 23 minutes to solve scenario M15-BAL-L (in M15-BAL-L the solution is on average 1% larger than the best found 189). These times will be reduced by using the proposed algorithm in a problem decomposition approach, as shown in Section 4.4.

4.4 Benefit of Problem Decomposition

In the previous section, we show that the proposed generative method can reduce the effort of the evaluative method, i.e., the simulation budget. In this

section, we show that the use of the problem decomposition framework can reduce the search effort in the proposed generative method.

In this section, the algorithms are compared in terms of solution obtained and the required computational time. Time is used instead of simulation budget for two reasons. One is that within the decomposition framework, the simulation budget used at a certain hierarchy is not equivalent to that used at a higher one, which makes it difficult to compare the simulation effort for different hierarchies. The other reason is that the use of the problem decomposition approach has a significant effect on algorithm efficiency because of the time spent in the generative method, which does not affect the simulation budget. As in Section 4.3, the simulated sample-path used to evaluate solution $\mathbf{x}$ is fixed for each scenario. Results of this section have been obtained with Matlab2018 on a server with Xeon cores and 196 GB of RAM (data refers to a single core).

GA parameters are as in Table 2, except that the population size is selected as $\min(10 * \ell, 50)$ for hierarchy ℓ. Algorithm parameters (initial budget n_0 and EI_{target}) have been tuned according to problem dimension (i.e., the hierarchy ℓ) as in Table 4. It is important to mention that the final problem (i.e., the system belonging to the higher hierarchy $\ell = S-1$) is solved with $EI_{\text{target}} = 0$ to have a more accurate solution, as discussed in Section 4.3.1.

Table 4 Selected initial budget size and stopping criterion for each sub-problem $M(\ell, j)|\ell = 1, \ldots, S-2$ where $\mathbf{x}^{\text{best}}_{(\ell,j)}$ is the current best solution for the sub-problem $M(\ell, j)$.

Algorithm	Initial Budget Size n_0	Sub-problem EI_{target}
Dec + KR	$5 * \ell$	$2\% * z(\mathbf{x}^{\text{best}}_{(\ell,j)})$
Dec + EKR	$3 * \ell$	$8\% * z(\mathbf{x}^{\text{best}}_{(\ell,j)})$ for M5 scenarios $0.2\% * z(\mathbf{x}^{\text{best}}_{(\ell,j)})$ for M15 scenarios

4.4.1 The Effect of Problem Decomposition and Throughput Target

The average total computational time that different algorithm settings require to solve the BAP is reported in Table 5. Comparing the use of KR and EKR is aligned with Section 4.3: the use of DDX reduces the time required to find a BAP solution with up to an 85% reduction in the M5-MID-H scenario (51% on the average, only in the M5-B2-H scenario are the results not significantly different).

In the evaluated cases, the problem decomposition approach further reduces the computational time and it is more efficient when high total buffer capacity is required, i.e., "H" scenarios with a high throughput target. Indeed, the lower bounds set by low hierarchies in "H" scenarios are higher than those in "L" scenarios because the throughput target constraint is present in all the sub-problems. As a consequence, the remaining search space is small in "H" scenarios and the efficiency is highly improved. Compared to EKR, Dec+EKR

saves on average 83% of the time for high target scenarios and 48% for low target ones. Similar results apply for KR versus Dec+KR.

Scenario	$z(\mathbf{x}^*)$	Computational Time (Relative frequency of exact solution)			
		KR	Dec+KR	EKR	Dec+EKR
M5-BAL-H	63	100 ± 21 (1)	50 ± 7 (0.94)	65 ± 14 (1)	8 ± 0.3 (1)
M5-BAL-L	39	70 ± 15 (1)	68 ± 13 (0.84)	18 ± 6 (1)	11 ± 1 (1)
M5-MID-H	55	213 ± 43 (1)	59 ± 12 (0.92)	33 ± 7 (1)	11 ± 1 (1)
M5-MID-L	35	111 ± 28 (1)	105 ± 24 (0.94)	60 ± 19 (1)	30 ± 8 (1)
M5-B2-H	83	128 ± 26 (1)	31 ± 4 (0.98)	139 ± 27 (1)	9 ± 0.3 (1)
M5-B2-L	45	91 ± 18 (1)	67 ± 18 (0.5)	23 ± 5 (1)	10 ± 1 (0.98)

Table 5 The mean computational times [seconds] required to solve the BAP and the corresponding 95% confidence intervals according to the algorithm used (50 algorithm replications). The numbers in brackets represent the relative frequency of the algorithm finding the exact solution $z(\mathbf{x}^*)$.

It might happen that algorithms Dec+EKR and Dec+KR do not find the optimal solution. Indeed, the solution found might not be exact when $EI_{target} > 0$ (Section 4.3.1). As a consequence, a bound might cut the optimal solution. In Table 5, the numbers in brackets represent the relative frequency of the algorithm finding the exact solution $z(\mathbf{x}^*)$. As a consequence, solving sub-problem $M(\ell, j)$ with $EI^{(\ell,j)}_{\text{target}} > 0$ could save computational time, whereas, the lower the $EI^{(\ell,j)}_{\text{target}}$, the more accurate the bounds provided.

Figure 8 shows a more detailed comparison for scenarios M5-BAL-H and M5-BAL-L. Similarly to Figure 5, the evolution of the objective function is reported according to different algorithm settings, and computational time is used as the horizontal axis. Lines start when all replications reach a feasible solution for the system. Also, for Dec+KR and Dec+EKR, lines start when all sub-problems $M(\ell, j)$ are solved, which happens later compared to KR and EKR algorithms. The problem decomposition framework is efficient when high buffer capacities are needed, i.e., when the throughput target is high. On the contrary, if the throughput target is low, buffers are small and the benefit provided by the lower bounds is not significant and is counterbalanced by the additional effort required to solve sub-problems.

Similar results can be obtained with long lines: M15-BAL-H and M15-MID-H. We limit the comparison between algorithm settings Dec+EKR and EKR since the surrogate model created with KR is shown to be less efficient (cf. Section 4.3). As a reference, the *best found* solution among all algorithm replications is 291 for M15-BAL-H and 226 for M15-MID-H.

For M15-BAL-H, EKR stops after 18 hours, on average, and results in an average distance from the best found solution of 2.4 buffer spaces. Despite Dec+EKR starting to solve the last hierarchy after around 2 hours, it stops after 2.94 hours on average, and results in an average distance from the best found to be 1.15 buffer spaces. The advantage also appears in M15-MID-H: EKR and Dec+EKR stop after 11.7 and 3 hours respectively, and the average distance from the best found is 2.2 and 1.3 buffer slots respectively.

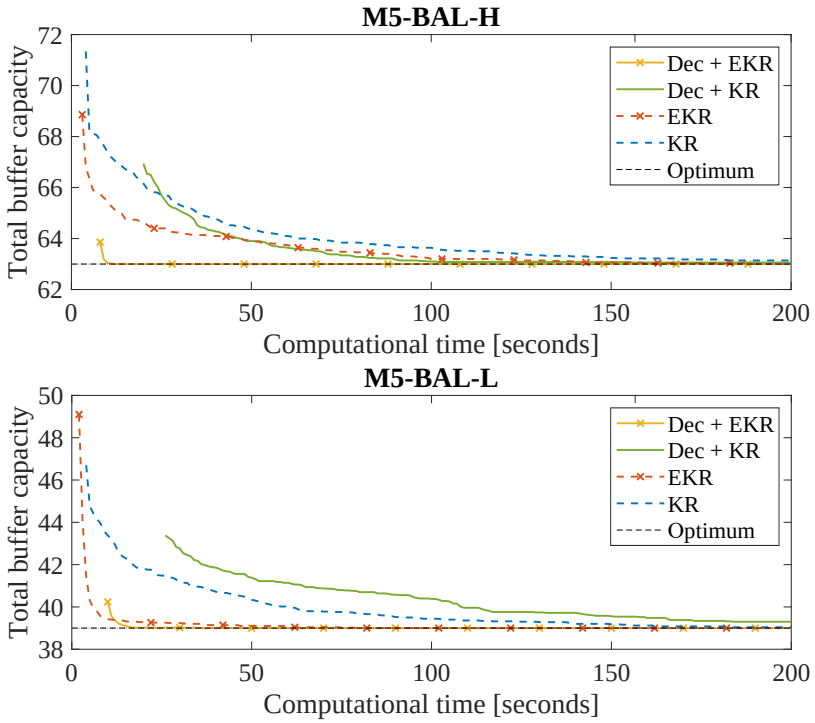

Fig. 8 Comparison of algorithm performance (mean of 50 algorithm replications) for scenarios M5-BAL-L and M5-BAL-H according to algorithm setting.

4.4.2 Jumping Approach

In the problem decomposition framework, the bounds applied reduce the solution space simplifying the search in the generative method. Table 6 shows the buffer allocation $\mathbf{x}^{\text{best}}_{(\ell,j)}$ found by solving BAP of sub-system $M(\ell, j)$. Solutions are optima and have been validated using the algorithm proposed in [Weiss and Stolletz, 2015]. $\mathcal{A}$ is used to denote the feasibility region of the final problem, i.e., $M(4, 1)$ having hierarchy $\ell = 4$ and including the whole system. We compute the fraction P of search space $\mathcal{A}$ that remains after all sub-systems are solved:

$$P = 1 - \sum_{\ell=1}^{S-2} \sum_{j=1}^{S-\ell} p_{(\ell,j)} \tag{14}$$

where $p_{(\ell,j)}$ is the fraction of additional cut space provided by the bound from sub-system $M(\ell, j)$. For the evaluated scenarios, the remaining spaces are 26%, 34%, and 7% of search space $\mathcal{A}$ respectively for M5-BAL-H, M5-MID-H, and M5-B2-H. For a low throughput target, the reduction is significant but much smaller, i.e., the remaining spaces are 70%, 66%, and 51% for M5-BAL-L, M5-MID-L, and M5-B2-L, respectively. Results in Table 6 support the results

obtained in Section 4.4.1: the bounds are more effective in "H" scenarios than in "L" scenarios.

	M5-BAL-H		M5-MID-H		M5-B2-H	
$M(\ell,j)$	$z(\mathbf{x}^{\text{best}}_{(\ell,j)})$	$p_{(\ell,j)}$	$z(\mathbf{x}^{\text{best}}_{(\ell,j)})$	$p_{(\ell,j)}$	$z(\mathbf{x}^{\text{best}}_{(\ell,j)})$	$p_{(\ell,j)}$
M(1,1)	6	0.19	1	0.03	11	0.35
M(1,2)	6	0.16	9	0.28	11	0.23
M(1,3)	6	0.13	9	0.20	11	0.15
M(1,4)	8	0.14	1	0.02	11	0.10
M(2,1)	22	0.03	22	0.06	28	0.01
M(2,2)	22	0.02	24	0.01	38	0.05
M(2,3)	24	0.03	22	0.04	28	0.00
M(3,1)	42	0.02	40	0.01	61	0.03
M(3,2)	44	0.02	39	0.01	60	0.01
	$z(\mathbf{x}^*)$	P	$z(\mathbf{x}^*)$	P	$z(\mathbf{x}^*)$	P
M(4,1)	63	0.26	55	0.34	83	0.07
	M5-BAL-L		M5-MID-L		M5-B2-L	
$M(\ell,j)$	$z(\mathbf{x}^{\text{best}}_{(\ell,j)})$	$p_{(\ell,j)}$	$z(\mathbf{x}^{\text{best}}_{(\ell,j)})$	$p_{(\ell,j)}$	$z(\mathbf{x}^{\text{best}}_{(\ell,j)})$	$p_{(\ell,j)}$
M(1,1)	1	0.03	1	0.03	3	0.10
M(1,2)	1	0.03	3	0.09	3	0.09
M(1,3)	1	0.03	3	0.08	3	0.08
M(1,4)	1	0.03	1	0.03	3	0.07
M(2,1)	12	0.05	12	0.03	16	0.05
M(2,2)	12	0.04	14	0.03	18	0.05
M(2,3)	14	0.06	12	0.03	16	0.03
M(3,1)	25	0.02	25	0.01	31	0.01
M(3,2)	26	0.01	24	0.01	31	0.01
	$z(\mathbf{x}^*)$	P	$z(\mathbf{x}^*)$	P	$z(\mathbf{x}^*)$	P
M(4,1)	39	0.70	35	0.66	45	0.51

Table 6 Effect of bounds on high and low-target scenarios. Solution $\mathbf{x}^{\text{best}}_{(\ell,j)}$ of each sub-problem $M(\ell,j)$ and fraction $p_{(\ell,j)}$ of the additional cut space. The fraction P of remaining search space for the final problem $M(4,1)$ is also reported.

From results in Table 6, the additional cut space $p_{(\ell,j)}$ reduces as the hierarchy ℓ of the sub-problem increases. This means that the benefit provided by solving a sub-problem reduces while approaching the final problem. Computational time to solve M15-BAL-H using the Dec+EKR algorithm and the remaining space given the lower bounds are analyzed as in Figure 9:

- A bowl effect is noticed in the computational time required to solve sub-systems in hierarchy ℓ.
- The fraction of the additional cut space over the whole domain (as for the fraction of the additional cut space over the remaining space) decreases as the hierarchy increases.

As the hierarchy increases, the effort required to solve all sub-problems is high whereas the benefit provided by the bounds is small. Moreover, the higher the hierarchy, the closer the created bound to the optimum. Thus, the risk of excluding the optimal solution from the search space increases as we approach the final problem. Hence, we investigate a "jumping strategy" that

only partially solves the set of sub-problems and focuses on those with lower hierarchies.

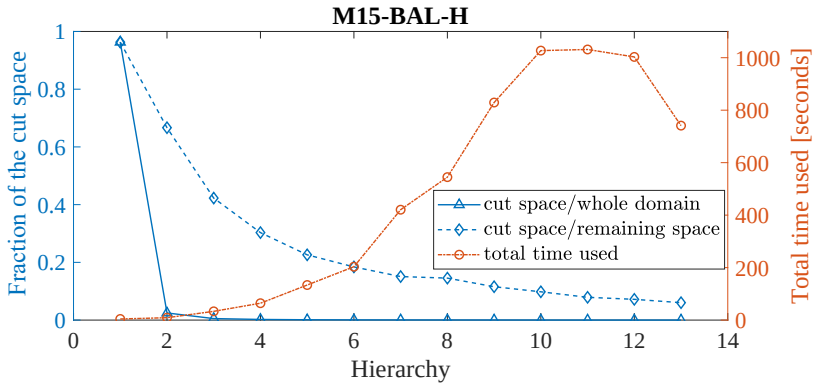

Fig. 9 Computational effort and efficiency of bounds for scenario M15-BAL-H. The fraction of the additional space cut by solving sub-problems at each hierarchy and the average of the total time used for solving sub-problems at each hierarchy for scenario M15-BAL-H. The diamond mark indicates that the fraction relates to the whole feasible domain and the triangular mark indicates the fraction with respect to the remaining space at the current hierarchy. The mean of 20 algorithm replications is reported.

A first analysis is focused on M5-BAL-H where several "jumping strategies" are evaluated. If all hierarchies are jumped, the algorithm solves the final problem directly (i.e., hierarchy $\ell = 4$) resulting in the EKR algorithm setting. Vice versa, if all hierarchies are executed (i.e., hierarchy $\ell = 1, 2, 3, 4$), the algorithm has Dec+EKR setting and results are equal to that of Section 4.4.1. We consider three "jumping" strategies: to solve hierarchies $\ell = \{1, 4\}$, to solve hierarchies $\ell = \{1, 2, 4\}$, and to solve hierarchies $\ell = \{1, 3, 4\}$. As shown in Figure 10, solving the first and last hierarchies ($\ell = \{1, 4\}$) is the best approach analyzed.

The benefit provided by lower bounds might be offset by the effort spent on solving sub-problems at high hierarchies. This phenomenon is more significant in long lines. For scenarios M15-BAL-H and M15-MID-H respectively, Figure 11 and Figure 12 show the solution provided by algorithms with different jumping strategies as the computational time increases. It can be found that, for cases for which a high total buffer capacity required (i.e., high throughput target), a decomposition approach is efficient compared to solving the final problem directly. Nevertheless, a long computational time is needed to solve sub-problems at all hierarchies before obtaining a feasible solution. As discussed for 5-machine lines, jumping high-hierarchies might save computational times.

The idea of skipping some sub-problems also appears in the segmentation approach proposed by Shi and Gershwin [Shi and Gershwin, 2016]. However, the authors do not apply a hierarchical solving approach, but focus on solving

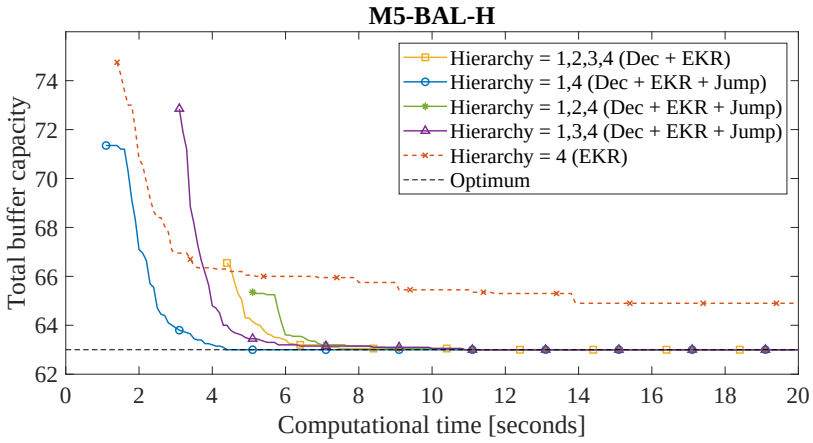

Fig. 10 Comparison of algorithm performance for scenarios M5-BAL-H (mean of 20 algorithm replications). The EKR model is used in decomposition framework although only a subset of hierarchies is solved.

some sub-problems and combining their local solutions. Therefore, the authors decompose the system into a few overlapping sub-systems and solve each of them to obtain local solutions.

Differently, in the proposed "jumping strategy", we set out to solve all sub-problems in a certain hierarchy because we cannot state a-priori which sub-problem is more significant in terms of generated cut without additional information.

4.4.3 The Effect of Re-using Data

The use of the EKR method to create the surrogate model combined with a decomposition approach enables the re-use of data from one hierarchy to the next. This feature might be useful when an analytical approach is not available.

M5-BAL-H is used as an example to evaluate how the re-use of data affects the results. Figure 13 shows that Dec + EKR (re-use) performs better than Dec + KR, although it is not as useful as including the DDX method. The computational effort required by the setting Dec + EKR (re-use) to solve the BAP is reported in Table 7 and can be compared with that for other settings (cf. Table 5). When helpful analytical methods cannot be applied or are not available, re-using data from lower hierarchical systems can also be promising compared to KR, that uses only HF simulation data.

This result is also supported by Figure 14 representing the prediction error obtained while evaluating the productivity of the final system $M(S - 1, 1)$. Three algorithm settings are compared using $n_c = 10^4$ checkpoints: Dec+KR, Dec+EKR(DDX), and Dec+EKR(re-use). When the proposed algorithm reaches the highest hierarchy and creates the surrogate model of the final system, the $MAPE$ is computed. For comparison purposes, the initial

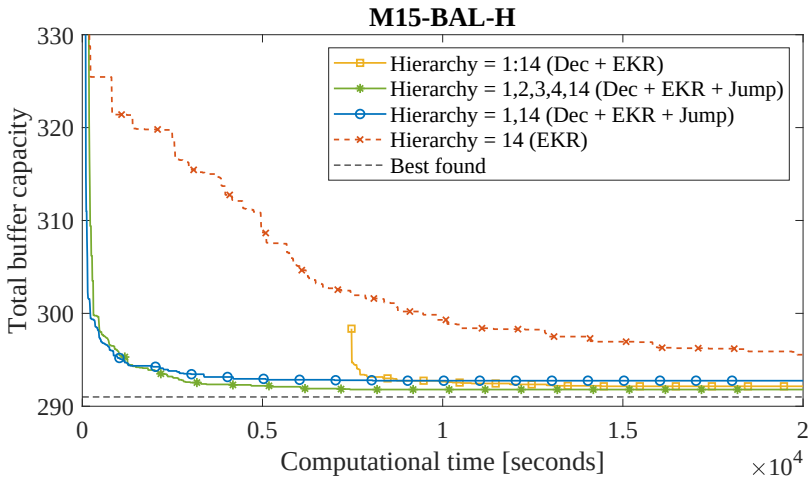

Fig. 11 Comparison of algorithm performance for scenario M15-BAL-H (mean of 20 algorithm replications).

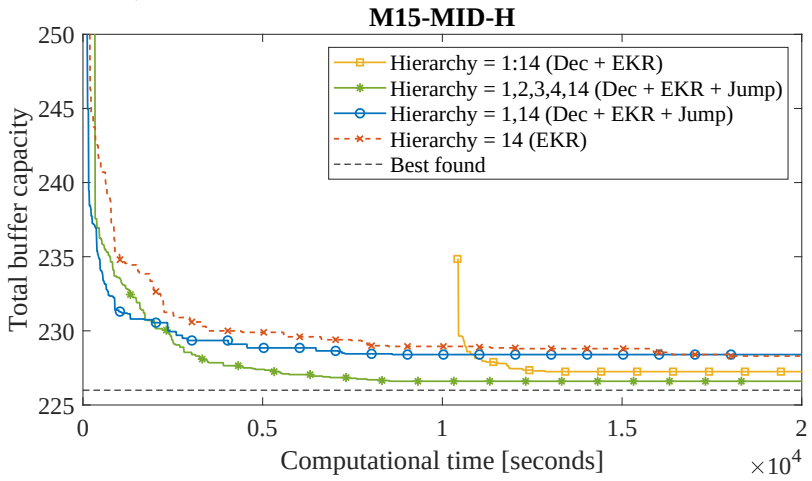

Fig. 12 Comparison of algorithm performance for scenario M15-MID-H (mean of 20 algorithm replications).

design at hierarchy $\ell = S - 1$ contains the same number of design points for different algorithm settings: $||\mathbb{X}_0|| = 16$ for scenario M5-BAL-H and $||\mathbb{X}_0|| = 56$ for scenario M15-BAL-L. These design points are sampled in the search space resulting after the cuts provided in the decomposed approach.

The surrogate model created using the EKR(DDX) has the smallest prediction error and re-using data also improves the quality of the built surrogate model compared to that created using the KR method.

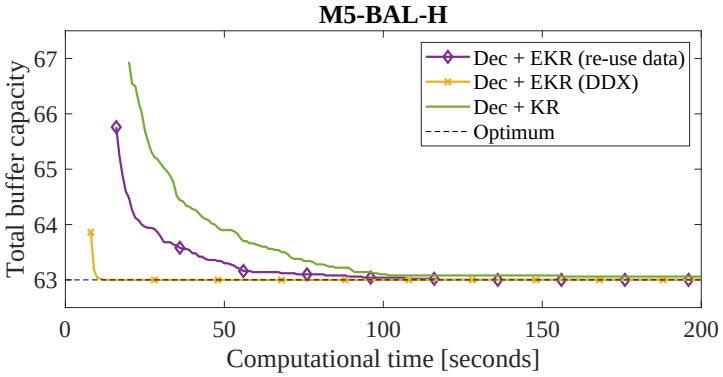

Fig. 13 Comparison of algorithm performance for scenarios M5-BAL-H (mean 50 algorithm replications).

Scenario	$z(\mathbf{x}^*)$	Computational Time	Relative frequency of exact solution
M5-BAL-H	63	39 ± 7	(1)
M5-BAL-L	39	57 ± 8	(1)
M5-MID-H	55	50 ± 9	(0.98)
M5-MID-L	35	72 ± 9	(1)
M5-B2-H	83	44 ± 10	(1)
M5-B2-L	45	56 ± 10	(0.94)

Table 7 The mean computational times [seconds] required to solve the BAP and the corresponding 95% confidence intervals with setting Dec+EKR(re-use) (50 algorithm replications). Relative frequency of the exact solution $z(\mathbf{x}^*)$ is also reported. Results can be compared with those of Table 5.

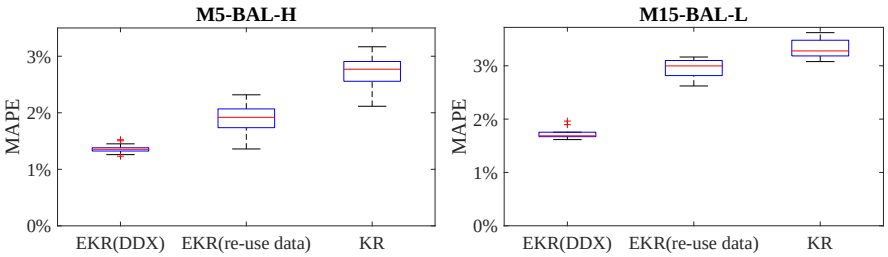

Fig. 14 *MAPE* of the surrogate models created by KR, EKR using DDX as LF, and EKR using lower hierarchy models as LF. Boxplots are created with 50 algorithm replications for scenario M5-BAL-H and 10 replications for scenario M15-BAL-L.

5 Conclusions

The proposed method is efficient when the evaluative method is time-consuming (e.g. highly detailed simulation model). Also, the algorithm has a significant advantage in BAP where the optimal total buffer capacity is high. These advantages are due to two properties of our approach:

1. The simulation budget is saved through an efficient allocation of the budget in near-optimum areas thanks to the use of the surrogate model;

2. The generative method increases efficiency as more search space is cut by the decomposition approach.

Because of the randomness of the initial design, and the GA used in the generative method, the overall algorithm has a heuristic nature. Nevertheless, the variability of the results is very limited. Further, the combination of simulation and analytical methods improves the accuracy of the surrogate model even with few observations and improves the efficiency of the algorithm significantly. Where no analytical method is available, re-using data in the decomposition framework could be also helpful.

In the decomposition framework, running all the hierarchies might be not efficient. A trade-off exists between additional computational effort required to solve sub-problems and the size of solution space cut by the provided bounds. In this case, jumping some hierarchies might be helpful.

The proposed method is highly flexible in terms of applications besides the BAP. It can be applied potentially to other problems in which the decision variables are continuous or discrete values with a larger candidate set (so that the surrogate model can be built) and lower/upper bounds can be provided from lower hierarchies (so that the decomposition framework is helpful), e.g. resource and server allocation problems, line balancing problems, redundancy problems, design and control problems of production lines. Hence, future developments will be focused on generalising the approach.

Future work will be devoted to investigating the use of surrogate models in an exact approach. Moreover, "jumping" strategies will be further investigated in order to understand which hierarchies should be jumped to improve efficiency. Also, sub-systems containing the bottleneck can be prioritized compared to others since they provide more efficient bounds.

The EKR method provides an efficient way to create a surrogate model of system performance from multi-fidelity sources. The expensive high-fidelity data (e.g. outputs of highly detailed simulation models or data from the field) is combined with low-fidelity estimates (e.g. outputs of coarse simulation models or analytical methods), which are fast and easy to calculate, to improve the prediction performance of the built surrogate model. The EKR method can assign different weights to different low-fidelity models in different areas of the domain automatically, according to the observed data.

This section describes briefly how to build the surrogate model using the EKR method. A Matlab EKR toolbox (both the EKR code and a manual) can be found at [Lin et al., 2020] DOI: 10.13140/RG.2.2.16632.19206. More details about the method can be found in [Lin et al., 2019].

Given a system configuration $\boldsymbol{x}$ under which the high-fidelity system performance $y_h(\boldsymbol{x})$ is unknown, its low-fidelity outputs $y_{l_j}(\boldsymbol{x}), \forall j$ are firstly corrected by scaling functions. Two scaling functions are considered:

1. Additive scaling function: $\tilde{y}_i^{l_j}(\boldsymbol{x}) = y_{l_j}(\boldsymbol{x}) + (y_h(\boldsymbol{x}_i^0) - y_{l_j}(\boldsymbol{x}_i^0)), \forall i \in \mathcal{N}, \forall j \in \mathcal{J}$;
2. Multiplicative scaling function: $\tilde{y}_i^{l_j}(\boldsymbol{x}) = \frac{y_h(\boldsymbol{x}_i^0)}{y_{l_j}(\boldsymbol{x}_i^0)} \cdot y_{l_j}(\boldsymbol{x}), \forall i \in \mathcal{N}, \forall j \in \mathcal{J}$.

where $y_h(\boldsymbol{x}_i^0)$ and $y_{l_j}(\boldsymbol{x}_i^0)$ are the outputs of the high-fidelity model and the j-th low-fidelity model at the i-th design point $\boldsymbol{x}_i^0$, respectively. Different scaling functions can be used for different low-fidelity models.

These corrected outputs are expected to have more reliable prediction performance if their scaling functions are estimated by the initial design points close to the unobserved point. Therefore, for the j-th low-fidelity model, Kernel regression [Wand and Jones, 1995] is used to locally fit a polynomial on the corrected data $\tilde{y}_i^{l_j}(\boldsymbol{x})$ with distance-based weights. The system performance estimate at the unobserved point $\boldsymbol{x}$, using the j-th low-fidelity model's corrected data, has a closed form:

$$\hat{y}_{l_j}(\boldsymbol{x}) = \boldsymbol{e}_1^T(\boldsymbol{X}_{\boldsymbol{x}}^T\boldsymbol{W}_{\boldsymbol{x}}\boldsymbol{X}_{\boldsymbol{x}})^{-1}\boldsymbol{X}_{\boldsymbol{x}}^T\boldsymbol{W}_{\boldsymbol{x}}\tilde{\boldsymbol{Y}}_{l_j}, \forall j \in \mathcal{J}, \tag{15}$$

where $\boldsymbol{e}_1$ is a $(dp+1)$-dimensional vector whose first element is 1 and the rest are 0,

$$\boldsymbol{X}_{\boldsymbol{x}} = \begin{bmatrix} 1 & (\boldsymbol{x}_1^0 - \boldsymbol{x})^T & \cdots & \left[(\boldsymbol{x}_1^0 - \boldsymbol{x})^p\right]^T \\ 1 & (\boldsymbol{x}_2^0 - \boldsymbol{x})^T & \cdots & \left[(\boldsymbol{x}_2^0 - \boldsymbol{x})^p\right]^T \\ \vdots & \vdots & \ddots & \vdots \\ 1 & (\boldsymbol{x}_n^0 - \boldsymbol{x})^T & \cdots & \left[(\boldsymbol{x}_n^0 - \boldsymbol{x})^p\right]^T \end{bmatrix}$$

is an $n \times (dp+1)$ matrix, p is the degree of the fitted polynomial and $\tilde{\boldsymbol{Y}}_{l_j} = [\tilde{y}_1^{l_j}(\boldsymbol{x}), \cdots, \tilde{y}_n^{l_j}(\boldsymbol{x})]^T$. $\boldsymbol{W}_{\boldsymbol{x}} = \text{diag}\{K_{1,\boldsymbol{\Theta}_1}(\boldsymbol{x}_1^0 - \boldsymbol{x}), \cdots, K_{1,\boldsymbol{\Theta}_1}(\boldsymbol{x}_n^0 - \boldsymbol{x})\}$ is an $n \times n$ diagonal matrix where

$$K_{1,\boldsymbol{\Theta}_1}(\boldsymbol{x}_i^0 - \boldsymbol{x}) = \prod_{k=1}^{d} \exp\left\{-\frac{1}{2\theta_{1,k}}(x_{ik}^0 - x_k)^2\right\}, \forall i \in \mathcal{N},$$

$\boldsymbol{\Theta}_1 = \text{diag}\{\theta_{1,1}, \cdots, \theta_{1,d}\}$, and $\theta_{1,k} > 0, k = 1, \cdots, d$ are parameters to be selected.

Finally, the estimates from different low-fidelity models are combined with the weights related to the estimated weighted square error:

$$\hat{y}_{\text{EKR}}(\boldsymbol{x}) = \sum_{j \in \mathcal{J}} w_{l_j}(\boldsymbol{x})\hat{y}_{l_j}(\boldsymbol{x}),$$

where:

$$w_{l_j}(\boldsymbol{x}) = \frac{K_{2,\theta_2}(\hat{WSE}_{l_j}(\boldsymbol{x}))}{\sum_{i \in \mathcal{J}} K_{2,\theta_2}(\hat{WSE}_{l_i}(\boldsymbol{x}))},$$

$$\hat{WSE}_{l_j}(\boldsymbol{x}) = (\text{tr}(\boldsymbol{W}_{\boldsymbol{x}}))^{-1}\tilde{\boldsymbol{Y}}_{l_j}^T(\boldsymbol{W}_{\boldsymbol{x}} - \boldsymbol{W}_{\boldsymbol{x}}^T\boldsymbol{X}_{\boldsymbol{x}}(\boldsymbol{X}_{\boldsymbol{x}}^T\boldsymbol{W}_{\boldsymbol{x}}\boldsymbol{X}_{\boldsymbol{x}})^{-1}\boldsymbol{X}_{\boldsymbol{x}}^T\boldsymbol{W}_{\boldsymbol{x}})\tilde{\boldsymbol{Y}}_{l_j}, \forall j \in \mathcal{J},$$

and $\text{tr}(\boldsymbol{W}_{\boldsymbol{x}})$ is the trace of $\boldsymbol{W}_{\boldsymbol{x}}$. $K_{2,\theta_2}(\cdot)$ has the following form:

$$K_{2,\theta_2}(\hat{WSE}_{l_j}(\boldsymbol{x})) = \exp\left\{-\frac{\hat{WSE}_{l_j}(\boldsymbol{x})}{2\theta_2 WSE_{min}(\boldsymbol{x})}\right\}, \forall j \in \mathcal{J},$$

where:

$$WSE_{min}(\boldsymbol{x}) = \min_{j\in\mathcal{J}}\{\hat{WSE}_{l_j}(\boldsymbol{x})\},$$

and θ_2 is an unknown parameter to be selected.

The model parameters $\theta_{1,k}, \theta_2$ are selected according to the cross validation. Where $p = 1$, the estimated root square error of $\hat{y}_{\mathrm{EKR}}(\boldsymbol{x})$ as an estimate of the high-fidelity response is provided as:

$$\hat{s}_{\mathrm{EKR}}(\boldsymbol{x}) = \sqrt{\hat{WSE}(\boldsymbol{x})\left(1 + \frac{1}{2^{d/2}\mathrm{tr}(\boldsymbol{W_x})}\right)}$$

where:

$$\hat{WSE}(\boldsymbol{x}) = (\mathrm{tr}(\boldsymbol{W_x}))^{-1}\tilde{\boldsymbol{Y}}^T(\boldsymbol{W_x} - \boldsymbol{W_x}^T\boldsymbol{X_x}(\boldsymbol{X_x}^T\boldsymbol{W_x}\boldsymbol{X_x})^{-1}\boldsymbol{X_x}^T\boldsymbol{W_x})\tilde{\boldsymbol{Y}}$$

and:

$$\tilde{\boldsymbol{Y}} = \left[\sum_{j\in\mathcal{J}} w_{l_j}(\boldsymbol{x})\tilde{y}_1^{l_j}(\boldsymbol{x}), \cdots, \sum_{j\in\mathcal{J}} w_{l_j}(\boldsymbol{x})\tilde{y}_n^{l_j}(\boldsymbol{x})\right]^T.$$

References

[Alfieri and Matta, 2012] Alfieri, A. and Matta, A. (2012). Mathematical programming formulations for approximate simulation of multistage production systems. *European Journal of Operational Research*, 219(3):773–783.

[Colledani and Gershwin, 2013] Colledani, M. and Gershwin, S. (2013). A decomposition method for approximate evaluation of continuous flow multi-stage lines with general markovian machines. *Annals of Operations Research*, 209(1):5–40.

[Dallery et al., 1988] Dallery, Y., David, R., and Xiaolan, X. (1988). An efficient algorithm for analysis of transfer lines with unreliable machines and finite buffers. *IIE Transactions*, 20:280–283.

[Dolgui et al., 2007] Dolgui, A., Eremeev, A. V., and Sigaev, V. S. (2007). Hbba: Hybrid algorithm for buffer allocation in tandem production lines. *Journal of Intelligent Manufacturing*, 18(3):411–420.

[Frigerio et al., 2018] Frigerio, N., Lin, Z., and Matta, A. (2018). Multi-fidelity models for decomposed simulation optimization. In Rabe M. et al., editor, *Proceedings of the 2018 Winter Simulation Conference (WSC)*, pages 2217–2248, Gothenburg, Sweden.

[Gershwin, 1987] Gershwin, S. B. (1987). An efficient decomposition method for the approximate evaluation of tandem queues with finite storage space and blocking. *Operations Research*, 35(2):291–305.

[Gershwin and Schor, 2000] Gershwin, S. B. and Schor, J. E. (2000). Efficient algorithms for buffer space allocation. *Annals of Operations Research*, 93(1-4):117–144.

[Helber et al., 2011] Helber, S., Schimmelpfeng, K., Stolletz, R., and Lagershausen, S. (2011). Using linear programming to analyze and optimize stochastic flow lines. *Annals of Operations Research*, 182(1):193–211.

[Hillier, 2000] Hillier, M. S. (2000). Characterizing the optimal allocation of storage space in production line systems with variable processing times. *IIE Transactions*, 32(1):1–8.

[Jones et al., 1998] Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492.

[Kose and Kilincci, 2015] Kose, S. Y. and Kilincci, O. (2015). Hybrid approach for buffer allocation in open serial production lines. *Computers & Operations Research*, 60:67–78.

[Law and Kelton, 2000] Law, A. M. and Kelton, W. D. (2000). *Simulation Modeling & Analysis*. McGraw-Hill, Inc, New York, 3rd edition.

[Li and Meerkov, 2009] Li, J. and Meerkov, S. (2009). *Production systems engineering*. cited By 292.

[Liberopoulos et al., 2006] Liberopoulos, G., Papadopoulos, C., Tan, B., MacGregor Smith, J., and Gershwin, S. (2006). *Stochastic modeling of manufacturing systems: Advances in design, performance evaluation, and control issues*. cited By 5.

[Lin et al., 2020] Lin, Z., Matta, A., and Shanthikumar, J. (2020). A matlab extended kernel regression toolbox, doi: 10.13140/rg.2.2.16632.19206.

[Lin et al., 2019] Lin, Z., Matta, A., and Shanthikumar, J. G. (2019). Combining simulation experiments and analytical models with area-based accuracy for performance evaluation of manufacturing systems. *IISE Transaction*, 51(3):266–283.

[Matta, 2008] Matta, A. (2008). Simulation optimization with mathematical programming representation of discrete event systems. In S. J. Mason et al., editor, *Proceedings of the 2008 Winter Simulation Conference*, pages 1393–1400, Miami, FL. IEEE.

[Matta et al., 2012] Matta, A., Pezzoni, M., and Semeraro, Q. (2012). A kriging-based algorithm to optimize production systems approximated by analytical models. *Journal of Intelligent Manufacturing*, 23(3):587–597.

[McKay et al., 1979] McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245.

[Mockus et al., 1978] Mockus, J., Tiesis, V., and Zilinskas, A. (1978). The application bayesian methods for seeking the extremum. *Towards Global Optimisation*, 2:117–129.

[Papadopoulos et al., 2009] Papadopoulos, C., OKelly, M., Vidalis, M., and Spinellis, D. (2009). *Analysis and Design of Discrete Part Production Lines*, volume 31.

[Sacks et al., 1989] Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science*, pages 409–423.

[Shi and Gershwin, 2016] Shi, C. and Gershwin, S. B. (2016). A segmentation approach for solving buffer allocation problems in large production systems. *International Journal of Production Research*, 54(20):6121–6141.

[Shi and Men, 2003] Shi, L. and Men, S. (2003). Optimal buffer allocation in production lines. *IIE Transactions*, 35(1):1–10.

[Soyster et al., 1979] Soyster, A. L., Schmidt, J., and Rohrer, M. (1979). Allocation of buffer capacities for a class of fixed cycle production lines. *AIIE Transactions*, 11(2):140–146.

[Stolletz and Weiss, 2013] Stolletz, R. and Weiss, S. (2013). Buffer allocation using exact linear programming formulations and sampling approaches. *IFAC Proceedings Volumes*, 46(9):1435–1440.

[Tolio and Matta, 1998] Tolio, T. and Matta, A. (1998). A method for performance evaluation of automated flow lines. *CIRP Annals - Manufacturing Technology*, 47(1):373–376. cited By 55.

[Wand and Jones, 1995] Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*. Monographs on Statistics & Applied Probability. Chapman and Hall/CRC, New York.

[Weiss et al., 2018] Weiss, S., Matta, A., and Stolletz, R. (2018). Optimization of buffer allocations in flow lines with limited supply. *IISE Transactions*, 50(3):191–202.

[Weiss et al., 2019] Weiss, S., Schwarz, J. A., and Stolletz, R. (2019). The buffer allocation problem in production lines: Formulations, solution methods, and instances. *IISE Transactions*, 51(5):456–485.

[Weiss and Stolletz, 2015] Weiss, S. and Stolletz, R. (2015). Buffer allocation in stochastic flow lines via sample-based optimization with initial bounds. *OR Spectrum*, 37(4):869–902.