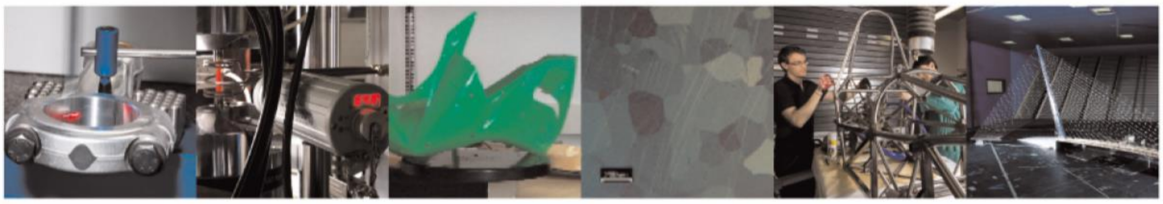
POLITECNICO
MILANO 1863

DIPARTIMENTO DI MECCANICA
mecc



Buffer allocation problem in production flow lines: A new Benders-decomposition-based exact solution approach

Mengyi Zhang, Erica Pastore, Arianna Alfieri, Andrea Matta

This is an Accepted Manuscript of an article published by Taylor & Francis in [JOURNAL TITLE]IISE Transactions on 22 Mar 2021, available online: https://www.tandfonline.com/doi/full/10.1080/24725854.2021.1905195.

This content is provided under CC BY-NC-ND 4.0 license

DIPARTIMENTO DI MECCANICA ■ POLITECNICO DI MILANO
via G. La Masa, 1 ■ 20156 Milano ■ EMAIL (PEC): pecmecc@cert.polimi.it
http://www.mecc.polimi.it
Rev. 0

Buffer Allocation Problem in production flow lines: a new Benders-decomposition-based exact solution approach

Mengyi Zhang [a], Erica Pastore [b], Arianna Alfieri [b] and Andrea Matta [a]
[a] Department of Mechanical Engineering, Politecnico di Milano, Milano, Italy
[b] Department of Management and Industrial Engineering,
Politecnico di Torino, Torino, Italy

Abstract

The buffer allocation problem (BAP) in production flow lines is very relevant from a practical point of view and very challenging from a scientific perspective. For this reason, it has drawn great attention both in industry and in the academic community. However, despite the problem relevance, no exact method is available in the literature to solve it when long production lines have to be dealt with, i.e., in practical settings. This work proposes a new Mixed Integer Linear Programming (MILP) formulation for exact solution of sample-based BAP. Due to the huge number of variables and constraints in the model, an algorithm based on Benders decomposition is proposed to increase the computational efficiency. The algorithm iterates between a simulation module that generates the Benders cuts and an optimization module that involves the solution of an updated MILP model. Multiple Benders cuts after each simulation run are generated by exploiting the structural properties of reversibility and monotonicity of flow line throughput. The new MILP formulation is tighter than the state-of-the-art model from a theoretical point of view, and order of magnitude of computation time saving is also observed in the numerical results.

Keywords: Buffer allocation problem; Production flow lines; Benders decomposition; Design optimization; Manufacturing systems design; Integer programming.

1 Introduction

Flow lines are typical architectures of production systems in many fields such as food, automotive, and electronics industries (Tempelmeier, 2003; Li, 2013), thus making design of flow lines a relevant topic from an industrial perspective. This work addresses a specific problem in flow line design, i.e., the Buffer Allocation Problem (BAP) (Weiss et al., 2019). BAP aims at finding the optimal buffer spaces between production stages to optimize a buffer related performance measure (e.g., minimum buffer costs) and that is able to guarantee a requested threshold value of another performance measure influenced by buffers (e.g., throughput, service level).

Buffer allocation is critical in flow line design, as the system performance is sensitive to the buffer allocation due to various sources of variability. In industrial contexts, the variability can be associated with human operators, machine failures, heterogeneity of job sizes/surfaces, and it is represented by the randomness in the time to process jobs (Inman, 1999; Li et al., 2009). It has been largely documented that variability affects system flow time and, hence, system throughput. Intermediate buffers between machines allow to mitigate the effect of local variability throughout the entire production line. Specifically, they reduce the propagation of blocking and starvation occurrences, which usually negatively affect system performance (Gershwin, 1994). Roughly speaking, the larger the buffer capacity, the higher the expected throughput of the line. However, large buffer spaces (i.e., large space needed to accommodate the entire production line) generate high costs, both in terms of costs directly related to space and in terms of operational costs of having large work-in-process (WIP) and long delivery lead times of customer orders. Thus, the design of flow lines has to consider the trade-off between the performance and the cost.

The complexity of BAP can be recognized also from a scientific point of view. Due to the lack of closed-form performance evaluation approaches for multi-stage flow lines, the convexity of BAP is not proven yet. The non-convex aspect implies the existence of multiple local optima, while the integer aspect implies that even solving for a local optimum may require exponential computation time. Practical relevance and complexity have made BAP attractive to researchers for decades, and this work proposes a sample-path exact approach for BAP. A brief literature review on exact approaches among the vastly existing works is provided below, and the contribution of this work follows.

1.1 Literature review

In a recent comprehensive survey (Weiss et al., 2019), a lack of exact methods and limited capability of the few state-of-the-art exact methods to solve long flow lines can be noticed (in this paper, a method is defined as *exact* if it has both an exact performance evaluation and an exact objective function optimization). Among the 145 methods categorized in Weiss et al. (2019), approximate methods (meta-heuristics and approximate search algorithms) dominate the field, especially for handling large production lines. On the contrary, only 18 approaches are exact, and they are reviewed below.

Exact explicit solution procedures use analytic methods to evaluate the line performance with closed-form formulae. These approaches are usually developed under strict assumptions, and the only work applying an exact explicit solution procedure is Enginarlar et al. (2002), which deals with a 2-machine unreliable line with exponential failures and repairs.

Integrated optimization methods formalize the BAP into a mathematical programming model, in which the performance evaluation based on samples or analytic results are included, and solve the model to the optimum. Some works used sample-based optimization and formulated BAP with mixed integer linear programming (MILP) (Matta, 2008; Stolletz and Weiss, 2013; Matta et al., 2014). Their models may suffer from sampling errors from stochastic distributions, which is usually mitigated by increasing the sample size. Solving MILP to the optimum becomes computationally challenging as the stage number increases, and the largest instance in this category is the 5-machine line solved in Matta (2008).

Iterative optimization methods decouple the search for the optimal buffer allocation and the evaluation of system performance in two phases, which are repeated until an optimal solution is found. Exact iterative methods can be further categorized into enumeration or search methods. The majority of the state-of-the-art iterative exact methods used enumeration (Powell and Pyke, 1996; Andijani, 1998; Conway et al., 1988, e.g.). As candidate solutions grow exponentially, enumeration methods are extremely time-consuming. These methods tackled up to 8-machine lines (Powell and Pyke, 1996). Exact search methods (Nori and Sarker, 1998; Seo et al., 2009; Seo and Lee, 2011; Weiss and Stolletz, 2015; Weiss et al., 2018) are developed based on structural properties. For instance, the performance, as a function of buffer spaces, can be separable (Seo et al., 2009; Seo and Lee, 2011), or convex (Nori and Sarker, 1998) under some strict assumptions such as 2-machine line, specific distribution of processing time and/or arrival process. However, in general cases, these assumptions are not satisfied. A more general structural property is that performance, as a function of buffer capacity, is monotonic, i.e., the system throughput cannot be worsened if the capacity of at least one buffer is increased (if the capacity of other buffers is not reduced). This property holds for multi-stage flow line with general distribution of processing time, time between failure and time to repair of unreliable machines (Weiss and

Stolletz, 2015). This property is used to design the algorithm searching for the lower bound of the constrained minimization problem in Seo et al. (2009); Seo and Lee (2011); Weiss and Stolletz (2015); Weiss et al. (2018). Speeding up techniques are also studied; for instance, upper and lower bounds can be generated before or during the searching procedure, and the search can start from downstream buffers to allocate capacity. Search methods are usually faster than enumeration and integrated methods. By using searching methods, a 24-machine line was solved in Weiss and Stolletz (2015), which represents the longest line solved by exact methods. However, as MILPs need to be solved in search iterations, the computational budget to find the optimal solution still remains large.

Summarizing, exact approaches in the literature can be fast for 2-machine lines or for system with specific assumptions. For more general cases, exact approaches are either enumeration-based or based on mixed integer programming, both of which reflect that BAP is an NP-hard problem.

1.2 Contribution

The original contribution of this work is twofold: (1) it proposes a new efficient MILP formulation for sample-based BAP, and (2) it applies an improved Benders decomposition algorithm to exactly solve it.

This paper proposes a new model for BAP that improves the very first formulation in Matta (2008) under the same modeling framework of sample-based integrated simulation-optimization (Pedrielli et al., 2018). The proposed MILP differs from the state-of-the-art in the formulation of the big-M constraints related to buffer capacity, which are based on the flow line blocking mechanism. The resulting model can be significantly tighter than the state-of-the-art and some of the existing solution approaches, such as the search algorithm of Weiss and Stolletz (2015), can be directly applied on it. Thanks to tightness of the proposed model, the improvement on computational efficiency can be expected.

The proposed MILP model is hard to solve, and large computational effort is needed. Thus, Benders Decomposition (BD) (Benders, 1962), a decomposition-based cut-generation exact solution approach, is used and several techniques to improve the basic BD algorithm are developed. First, instead of directly solving the LP formulation of the BD sub-problem, a faster simulation-based approach is applied to generate cuts (Zhang and Matta, 2020).

Second, in each iteration, two feasibility cuts are derived, one from the original system and the other from the *reversed* system, based on the reversibility property of flow lines (Yamazaki et al., 1985). According to the reversibility property of production flow lines, the throughput of a system is equal to that of its reversed system (i.e., the system where parts flow from the last to the first machines), so the feasible region of the original system is the same as that of the reversed system. Thus, a feasibility cut generated from the reversed system is also a feasibility cut for the original system. This allows a reduction of the number of iterations needed to solve the problem to the optimum. Third, the cuts are improved to formulate a tighter linear relaxation of the BD master problem, which reduces the computation time to solve a single iteration of the master problem. Fourth, combinatorial cuts based on monotonicity of throughput on buffer capacity can also be applied each time an infeasible solution is visited. Finally, a cut selection heuristic is proposed to control the size of the model. Numerical analysis shows that the proposed framework can be order of magnitude faster than the state-of-the-art approach.

The remainder of the paper is organized as follows. Sections 2 and 3 present the new formulation and the solution procedure. Numerical results are devised in Section 4, and Section 5 concludes the paper with future research directions.

2 Mathematical models

2.1 Problem definition

The BAP addressed in this work finds the optimal buffer capacity of each stage in a flow line, i.e., the minimum buffer capacity able to guarantee the target throughput.

The type of flow line considered in the paper is depicted in Figure 1. It is composed of J machines and $J-1$ inter-machine buffers of finite capacity. All parts are processed by all the machines of the line, ordered according to the arrival sequence. Parts are always available in front of the first machine, and parts can immediately leave the system after being processed by the last machine. The inter-machine buffer j has finite capacity b_j. A full buffer causes the *blocking* of the upstream machine, and an empty buffer causes the *starvation* of the downstream machine. In this work, the *blocking-after-service* behavior

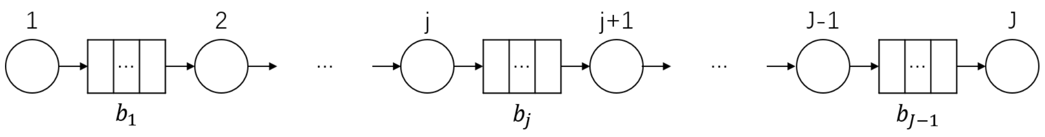


Figure 1: Flow line.

(BAS) is considered. The processing times of each machine can be generated as independent and identically distributed variables from a general distribution, or calculated as the sum of the two variables corresponding to the processing and the repair time, if the machine is considered unreliable. In the remainder of this paper, the term *processing time* will represent the processing time of reliable machines, and the processing time without including repair time of unreliable machines.

2.2 Notations and new formulation of BAP

Indices, sets and parameters of the proposed mathematical model are defined as follows.

$j \in \mathcal{J} = \{1, ..., J\}$	index, set and number of system stages
$\mathcal{J}^{'} = \{1, ..., J-1\}$	set of all the stages but the last one
$L_j, U_j, \ \forall \ j \in \mathcal{J}^{'}$	lower and upper bound of the capacity of buffer j
$i \in \mathcal{I} = \{1, ..., I\}$	index, set and number of parts in the simulation horizon
$t_{i,j}, \ \forall \ i \in \mathcal{I}, j \in \mathcal{J}$	processing time of part i at machine j
k	buffer capacity index
$\mathcal{K}_j = \{L_j + 1, ..., U_j\}, \forall \ j \in \mathcal{J}^{'}$	set of all the possible capacities for buffer b_j except the lower bound
$M_{i,j,k}, m_{i,j,k} \ \forall \ i \in \mathcal{I}, j \in \mathcal{J}^{'}, k \in \mathcal{K}_j$	big-M and small-M parameters
TH^*	target throughput

The proposed BAP mathematical model follows.

$$min \sum_{j \in \mathcal{J}^{'}} b_j$$

$$s.t. \qquad b_j = L_j + \sum_{k \in \mathcal{K}_j} y_{j,k} \qquad \forall j \in \mathcal{J}^{'} \tag{1}$$

$$y_{j,k} \leq y_{j,k-1} \qquad \forall j \in \mathcal{J}^{'},\ k \in \mathcal{K}_j \tag{2}$$

$$e^d_{i,j} - e^s_{i,j} \geq t_{i,j} \qquad \forall j \in \mathcal{J},\ i \in \mathcal{I} \tag{3}$$

$$e^s_{i,j} - e^d_{i-1,j} \geq 0 \qquad \forall j \in \mathcal{J},\ i \in \mathcal{I} \tag{4}$$

$$e^s_{i,j} - e^d_{i,j-1} \geq 0 \qquad \forall j \in \mathcal{J},\ i \in \mathcal{I} \tag{5}$$

$$e^d_{i,j} - e^s_{i-k,j+1} \geq -\sum_{k'=k+1}^{U_j} M_{i,j,k'} y_{j,k'} + \sum_{k'=L_j+1}^{k} m_{i,j,k'}(1 - y_{j,k'}) \qquad \forall j \in \mathcal{J}^{'},\ k \in \mathcal{K}_j, i \in \mathcal{I} \tag{6}$$

$$\frac{e^d_{I,J} - e^s_{1,1}}{I} \leq \frac{1}{TH^*} \tag{7}$$

$$y_{j,k} \in \{0, 1\} \qquad \forall j \in \mathcal{J}^{'},\ k \in \mathcal{K}_j \tag{8}$$

$$e^s_{i,j} \geq 0,\ e^d_{i,j} \geq 0 \qquad \forall j \in \mathcal{J},\ i \in \mathcal{I} \tag{9}$$

$$e^s_{i,j},\ e^d_{i,j},\ e^s_{1,1} = 0 \qquad \forall j \notin \mathcal{J}\ or\ i \notin \mathcal{I} \tag{10}$$

$$L_j \leq b_j \leq U_j,\ b_j \in \mathbb{Z}^+ \qquad \forall j \in \mathcal{J}^{'} \tag{11}$$

The integer variables b_j are the buffer capacity behind machine j. The objective function is the minimization of total buffer capacity. The 0-1 variables $y_{j,k}$ are used for buffer allocation, to set if capacity of buffer j is smaller or bigger than k. They are directly related to the buffer capacity b_j in the following way:

$$y_{j,k} = \begin{cases} 1, & \forall\ k \leq b_j \\ 0, & \forall\ k \geq b_j + 1. \end{cases} \tag{12}$$

Constraints (1) and (2) set $y_{j,k}$ value as requested by equation (12). It should be noticed that variables b_j can be calculated through equations (1) given the values of $y_{j,k}$, so they are actually redundant; however, they are kept in the model for simplicity of explanation.

Real-valued variables $e^s_{i,j}$ and $e^d_{i,j}$ represent the time part i enters and leaves machine j, respectively. Technically, indexes i and j are defined in sets $\mathcal{I}$ and $\mathcal{J}$, and for the cases $i \notin \mathcal{I}$ or $j \notin \mathcal{J}$ the values of $e^s_{i,j}$ and $e^d_{i,j}$ are both set to 0, as in (10). Constraints (3) to (7) evaluate the throughput and check if the target value is achieved. In particular, constraints (3) to (6) describe the *simulation trajectory* of the flow line based on input processing times $t_{i,j}$, and constraint (7) states that the throughput should reach the target value TH^*.

A *simulation trajectory* is the ordered set of occurring times of all the events collected during the run of a simulation model. For flow lines the definition is as follows.

Definition 2.1. Simulation trajectory of a flow line Given a flow line with buffer capacities $\bar{b}_j$ and sampled processing times $t_{i,j}$, the *simulation trajectory* is the collection of the event times $\bar{e}^s_{i,j}$ and $\bar{e}^d_{i,j}$ such that:

$$\bar{e}^s_{1,1} = 0 \tag{13}$$

$$\bar{e}^s_{i,j} = \max\{\bar{e}^d_{i-1,j}, \bar{e}^d_{i,j-1}\} \qquad i \in \mathcal{I},\ j \in \mathcal{J} \tag{14}$$

$$\bar{e}^d_{i,j} = \max\{\bar{e}^s_{i,j} + t_{i,j}, \bar{e}^s_{i-\bar{b}_j,j+1}\} \qquad i \in \mathcal{I},\ j \in \mathcal{J} \tag{15}$$

Within the simulation, buffer capacities $\bar{b}_j$ are parameters instead of variables, so they can be easily used in event time subscripts.

Equation (13) assures that the first part enters the first machine at time 0. Equations (14) define that a part can enter a machine only if both the part and the machine are available. These equations correspond to linear constraints (4)-(5) in the mathematical model. Equations (15) show that a part can leave a machine only if the processing is finished and there is an available space in the downstream buffer. These equations correspond to linear constraints (3) and (6). Specifically, constraints (6) cover all the k between L_j and U_j, and take into account the three occurrences that follow:

$$e^d_{i,j} - e^s_{i-\bar{b}_j,j+1} \geq 0 \qquad \forall i \in \mathcal{I},\ j \in \mathcal{J}' \tag{16}$$

$$e^d_{i,j} - e^s_{i-k,j+1} \geq -\sum_{k'=k+1}^{\bar{b}_j} M_{i,j,k'} \qquad \forall i \in \mathcal{I},\ j \in \mathcal{J}',\ k \leq \bar{b}_j - 1 \tag{17}$$

$$e^d_{i,j} - e^s_{i-k,j+1} \geq \sum_{k'=\bar{b}_j+1}^{k} m_{i,j,k'} \qquad \forall i \in \mathcal{I},\ j \in \mathcal{J}',\ k \geq \bar{b}_j + 1 \tag{18}$$

Given a buffer capacity $\bar{b}_j$ and its related 0-1 variables $\bar{y}_{j,k} = 1$, $\forall\, k \leq \bar{b}_j$ and $\bar{y}_{j,k} = 0$, $\forall\, k \geq \bar{b}_j + 1$, the right-hand side of constraints (6) becomes zero for $k = \bar{b}_j$, and constraints (6) are reduced to constraints (16). Similarly, for $k \leq \bar{b}_j - 1$, constraints (6) are reduced to constraints (17), and for $k \geq \bar{b}_j + 1$, constraints (6) are reduced to constraints (18). Constraints (16) represent the blocking occurrence due to the limited buffer capacity $\bar{b}_j$, as explained above. Moreover, constraints (17) and (18) should be redundant. If, for some i and j, because of the existence of (17) and (18), $\bar{e}^d_{i,j}$ is strictly greater than both

$\bar{e}^s_{i,j} + t_{i,j}$ and $\bar{e}^s_{i-\bar{b}_j,j+1}$, i.e., (15) is violated, the *simulation trajectory* will not be correctly modeled, and there may be bias on throughput evaluation in (7). The sufficient condition for equations (15) being correctly modeled by constraints (16), (17) and (18) is that (17) and (18) always hold for any simulation trajectory $\bar{e}^s_{i,j}$ and $\bar{e}^d_{i,j}$. Such violation may occur if a small value is set to $M_{i,j,k'}$, or a big value is set to $m_{i,j,k'}$. By simply setting $M_{i,j,k'} = +\infty$ and $m_{i,j,k'} = -\infty$, the formulation would be equivalent to that used in Matta (2008), which is proven to be weak. One of the original contributions of this work is the improvement of the state-of-the-art models by defining appropriate values for $M_{i,j,k'}$ and $m_{i,j,k'}$, which strengthen the model while keeping its correctness. The definition of bounding values for $M_{i,j,k'}$ and $m_{i,j,k'}$ from the input samples $t_{i,j}$ can be found in Propositions 2.2, 2.3 and 2.4, whose proofs can be found in Appendix I (online supplemental material).

Proposition 2.2. *In the simulation trajectory composed by the collection of event times $\bar{e}^s_{i,j}$ and $\bar{e}^d_{i,j}$ of the flow line with any buffer capacities $\bar{b}_j$ such that $L_j \le \bar{b}_j \le U_j$ based on samples $t_{i,j}$, if*

$$\bar{e}^s_{i+1,j} = \bar{e}^d_{i,j} \qquad \forall j \in \mathcal{J},\ i = 1, \dots, N-1, \tag{19}$$

i.e., if machine j does not undergo starvation just before processing part $i+1$ and after processing part i, then

$$\bar{e}^s_{i+1,j} - \bar{e}^s_{i,j} \le S_{i,j}, \tag{20}$$

where $S_{i,j}$ is the upper bound of the time interval needed for the next completion of a part after time $\bar{e}^s_{i,j}$ in the line segment from machine j till the end of the line. It can be formally defined as

$$S_{i,J} = t_{i,J},\ \forall\ i \in \mathcal{I} \tag{21}$$

$$S_{i,j} = \max\{t_{i,j}, t_{i_{j'},j'} | j' = j+1, ..., J, i_{j'} \in \mathcal{I}_{i,j,j'}\},\ \forall j \in \mathcal{J}',\ i \in \mathcal{I}. \tag{22}$$

Set $\mathcal{I}_{i,j,j'}$ includes all the parts in machine j' that could possibly block part $i+1$ in machine j when buffer capacities are varied between the lower and upper bounds. It can be mathematically defined as

$$\mathcal{I}_{i,j,j'} = \left[i - (j' - j) - \sum_{l=j}^{j'-1} U_l,\ i - (j' - j) - \sum_{l=j}^{j'-1} L_l \right] \cap \mathbb{Z}^+. \tag{23}$$

Proposition 2.3. *In the simulation trajectory composed by the collection of event times $\bar{e}^s_{i,j}$ and $\bar{e}^d_{i,j}$ of the flow line with any buffer capacities $\bar{b}_j$ such that $L_j \leq \bar{b}_j \leq U_j$ based on samples $t_{i,j}$, constraints* (17) *are always satisfied if*

$$M_{i,j,k'} = S_{i-k',j+1}, \tag{24}$$

$S_{i-k',j+1}$ is the upper bound of the time to complete the first part in the line segment downstream j after part $i-k'$ starts at $j+1$, i.e., $\bar{e}^s_{i-k',j+1}$, as in equations (21) *and* (22)*.*

Proposition 2.4. *In the simulation trajectory composed by the collection of event times $\bar{e}^s_{i,j}$ and $\bar{e}^d_{i,j}$ of the flow line with any buffer capacities $\bar{b}_j$ such that $L_j \leq \bar{b}_j \leq U_j$ based on samples $t_{i,j}$, constraints* (18) *are always satisfied if*

$$m_{i,j,k'} = t_{i-k',j+1}. \tag{25}$$

By setting the values of $M_{i,j,k'}$ and $m_{i,j,k'}$ as in equations (21)-(22), (24)-(25), the simulation trajectory of any specific buffer allocation is assured to be correctly represented by the model. Thus, using constraints (7), the throughput can be evaluated in a way equivalent to discrete-event simulation and the target throughput is surely achieved.

3 Benders-decomposition-based algorithm

The model previously proposed is a MILP model with a few integer variables but many real-value variables. Also, once the buffer capacities have been selected, the MILP reduces to an LP that represents the simulation trajectory of the flow line.

Those characteristics make Benders decomposition (Benders (1962)) a promising approaches for the problem, by using the master problem for the design aspect (i.e., the buffer allocation) and the sub-problem for throughput evaluation. Moreover, due to the *simulation nature* of the sub-problem, discrete event simulation is used for cut generation, as in Zhang and Matta (2020), instead of simplex or interior point methods, that are usually implemented in general-purpose LP solvers. The algorithm has linear complexity, in terms of computation time. After each simulation run, two classic Benders cuts are derived, one from the original system and the other from the reversed system. Also, each Benders cut can be easily improved to reach a tighter linear relaxation of the master problem. All the techniques are presented in the following.

3.1 Model decomposition and classic Benders cuts

The original model is partitioned in: a master problem (that includes the integer variables b_j and $y_{j,k}$) and a sub-problem (that includes the real-valued variables$e^s_{i,j}$ and $e^d_{i,j}$). The master problem is defined as follows:

$$min \sum_{j\in\mathcal{J}'} b_j \qquad s.t. \qquad (1),(2),(8),(11), feasibility\ cuts$$

As the objective function of the original complete model is included only in the master problem (as it is a function of the buffer capacity), the sub-problem is a feasibility problem, and only feasibility cuts (and not optimality cuts) are generated from its solution. Given the buffer capacity $\bar{b}_j$, the feasibility sub-problem is formulated as follows:

$$\begin{aligned}
&\min \varepsilon\\
s.t. \qquad & e^d_{i,j} - e^s_{i,j} \geq t_{i,j} \quad : u_{i,j} \quad \forall j\in\mathcal{J},\ i\in\mathcal{I} && (26)\\
& e^s_{i,j} - e^d_{i-1,j} \geq 0 \quad : v_{i,j} \quad \forall j\in\mathcal{J},\ i\in\mathcal{I} && (27)\\
& e^s_{i,j} - e^d_{i,j-1} \geq 0 \quad : s_{i,j} \quad \forall j\in\mathcal{J},\ i\in\mathcal{I} && (28)\\
& e^d_{i,j} - e^s_{i-\bar{b}_j,j+1} \geq 0 \quad : w_{i,j} \quad \forall j\in\mathcal{J},\ i\in\mathcal{I} && (29)\\
& \frac{e^d_{I,m}}{I} - \varepsilon \leq \frac{1}{TH^*} \quad : \vartheta && (30)\\
& e^s_{i,j} \geq 0,\ e^d_{i,j} \geq 0, \varepsilon \geq 0 \qquad \forall j\in\mathcal{J},\ i\in\mathcal{I}
\end{aligned}$$

where $u_{i,j}$, $v_{i,j}$, $s_{i,j}$, $w_{i,j}$ and ϑ are the dual variables, and ε is a non-negative variable to guarantee the feasibility. If the problem is feasible (i.e., there exists a buffer capacity configuration able to reach the target throughput), the optimal value of ε is zero.

Given the dual optimal solutions $\bar{u}_{i,j}$, $\bar{v}_{i,j}$, $\bar{s}_{i,j}$, $\bar{w}_{i,j}$ and $\bar{\vartheta}$, the feasibility Benders cut is:

$$-\sum_{\substack{j\in\mathcal{J}'\\ \bar{b}_j<U_j}} \sum_{k'=\bar{b}_j+1}^{U_j} \Big(\sum_{i\in\mathcal{I}} M_{i,j,k'}\bar{w}_{i,j}\Big) y_{j,k'} + \sum_{\substack{j\in\mathcal{J}'\\ \bar{b}_j>L_j}} \sum_{k'=L_j+1}^{\bar{b}_j} \Big(\sum_{i\in\mathcal{I}} m_{i,j,k'}\bar{w}_{i,j}\Big)(1-y_{j,k'}) + \sum_{i\in\mathcal{I},j\in\mathcal{J}} t_{i,j}\bar{u}_{i,j} - \bar{\vartheta}\frac{1}{TH^*} \leq 0 \qquad (31)$$

According to duality, $\sum_{i\in\mathcal{I},j\in\mathcal{J}} t_{i,j}\bar{u}_{i,j} - \bar{\vartheta}\frac{1}{TH^*} = \bar{\varepsilon}$, where $\bar{\varepsilon}$ is the solution of the primal sub-problem, and, hence, the cut can be re-written as:

$$-\sum_{\substack{j\in\mathcal{J}'\\ \bar{b}_j<U_j}}\sum_{k'=\bar{b}_j+1}^{U_j}\Big(\sum_{i\in\mathcal{I}}M_{i,j,k'}\bar{w}_{i,j}\Big)y_{j,k'}+\sum_{\substack{j\in\mathcal{J}'\\ \bar{b}_j>L_j}}\sum_{k'=L_j+1}^{\bar{b}_j}\Big(\sum_{i\in\mathcal{I}}m_{i,j,k'}\bar{w}_{i,j}\Big)(1-y_{j,k'})+\bar{\varepsilon}\leq 0 \quad (32)$$

The cut generation algorithm to derive the dual optimal solution was developed based on discrete event simulation as in Zhang and Matta (2020). They showed that the simulation-based cut generation algorithm can be orders of magnitude faster than general-purpose LP solvers. The algorithm is discussed in Appendix II (online supplemental material).

3.2 Tightening the Benders cut

The feasibility cuts (32) can be further improved to reach a better linear relaxation of the master problem as in the following proposition, whose proof can be found in Appendix I.

Proposition 3.1. *For each feasibility Benders cut* (32)*, a tighter valid inequality is:*

$$-\sum_{\substack{j\in\mathcal{J}'\\ \bar{b}_j<U_j}}\sum_{k'=\bar{b}_j+1}^{k_j}\Big(\sum_{i\in\mathcal{I}}M_{i,j,k'}\bar{w}_{i,j}\Big)y_{j,k'}+\sum_{\substack{j\in\mathcal{J}'\\ \bar{b}_j>L_j}}\sum_{k'=L_j+1}^{\bar{b}_j}\Big(\sum_{i\in\mathcal{I}}m_{i,j,k'}\bar{w}_{i,j}\Big)(1-y_{j,k'})+\bar{\varepsilon}\leq 0. \quad (33)$$

In the first term of equation (32)*, the second sum upper bound U_j is substituted by k_j, which is defined as follows: for all $j\in\mathcal{J}'$ and $\bar{b}_j<U_j$, parameter $\bar{b}_j+1\leq k_j\leq U_j$ is the minimal integer that makes the following inequality hold:*

$$-\sum_{k'=\bar{b}_j+1}^{k_j}\Big(\sum_{i\in\mathcal{I}}M_{i,j,k'}\bar{w}_{i,j}\Big)+\sum_{\substack{j'\in\mathcal{J}'\setminus\{j\}\\ \bar{b}_{j'}>L_{j'}}}\sum_{k'=L_{j'}+1}^{\bar{b}_{j'}}\Big(\sum_{i\in\mathcal{I}}m_{i,j',k'}\bar{w}_{i,j'}\Big)+\bar{\varepsilon}\leq 0. \quad (34)$$

If there is no value of k_j satisfying (34)*, then $k_j=U_j$.*

3.3 Reversed cut

A *reversed cut* is a feasibility cut similar to (32), but derived from the BAP of the *reversed flow line*. To formulate the BAP of the reversed flow line, namely the *reversed model*, a subscript transformation of both parameters and variables, as reported in Table 1, should be applied. The value of the redefined parameters is equal to the related ones in the original

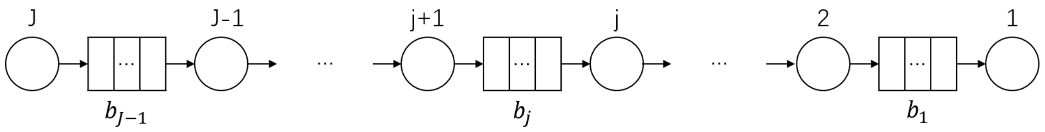


Figure 2: Reversed system.

model, e.g., $t^r_{I+1-i,J+1-j}$ is equal to $t_{i,j}$. The values of big-M and small-m parameters, instead, are recalculated by Propositions 2.2, 2.3 and 2.4, where $t_{i,j}$ are replaced by $t^r_{i,j}$. Generally speaking, $M_{i,j,k}$ ($m_{i,j,k}$) and $M^r_{I+1-i+k,J-j,k}$ ($m^r_{I+1-i+k,J-j,k}$) do not have the same value. The reversed model is as follows:

$$\begin{aligned}
&min \sum_{j\in\mathcal{J}'} b^r_j \\
s.t.\quad & b^r_j = L^r_j + \sum_{k\in\mathcal{K}^r_j} y^r_{j,k} && \forall j \in \mathcal{J}' \\
& y^r_{j,k} \le y^r_{j,k-1} && \forall j\in\mathcal{J}',\ k\in\mathcal{K}^r_j \\
& e^{dr}_{i,j} - e^{sr}_{i,j} \ge t^r_{i,j} && \forall j\in\mathcal{J},\ i\in\mathcal{I} \\
& e^{sr}_{i,j} - e^{dr}_{i-1,j} \ge 0 && \forall j\in\mathcal{J},\ i\in\mathcal{I} \\
& e^{sr}_{i,j} - e^{dr}_{i,j-1} \ge 0 && \forall j\in\mathcal{J},\ i\in\mathcal{I} \\
& e^{dr}_{i,j} - e^{sr}_{i-k,j+1} \ge -\sum_{k'=k+1}^{U^r_j} M^r_{i,j,k'} y^r_{j,k'} + \sum_{k'=L^r_j+1}^{k} m^r_{i,j,k'}(1-y^r_{j,k'}) && \forall j\in\mathcal{J}',\ k\in\mathcal{K}^r_j, i\in\mathcal{I} \\
& \frac{e^{dr}_{I,J} - e^{sr}_{1,1}}{I} \le \frac{1}{TH^*}
\end{aligned}$$

Redefined parameters	Primal variables	Dual variables
$t_{i,j}:\ t^r_{I+1-i,J+1-j}$	$e^s_{i,j}:\ -e^{dr}_{I+1-i,J+1-j}$	$u_{i,j}:\ u^r_{I+1-i,J+1-j}$
$U_j:\ U^r_{J-j}$	$e^d_{i,j}:\ -e^{sr}_{I+1-i,J+1-j}$	$v_{i,j}:\ v^r_{I+2-i,J+1-j}$
$L_j:\ L^r_{J-j}$	$b_j:\ b^r_{J-j}$	$s_{i,j}:\ s^r_{I+1-i,J+2-j}$
$\mathcal{K}_j:\ \mathcal{K}^r_{J-j}$	$y_{j,k}:\ y^r_{J-j,k}$	$w_{i,j}:\ w^r_{I+\bar{b}_{J-j}+1-i,J-j}$
Recalculated parameters		
$M_{i,j,k}:\ M^r_{I+1-i+k,J-j,k}$	$m_{i,j,k}:\ m^r_{I+1-i+k,J-j,k}$	

Table 1: Transformation of subscripts to develop the reversed model.

Provided that $\bar{b}^r_j = \bar{b}_{J-j}$ for any $j \in \mathcal{J}'$, and that all constraints containing big-M and small-m parameters are redundant, the sub-problem of the original model and that of the reversed model are equivalent (with the exception of the subscripts of the dual variables that are transformed as defined in Table 1) and, consequently, the dual optimal solutions are also equal. Therefore, a Benders cut of the reversed model, namely the *reversed cut*, can be generated after the simulation run of the original system, without repeating the dual variable calculation for the reversed system. The reversed cut is:

$$-\sum_{\substack{j\in\mathcal{J}'\\ \bar{b}^r_j<U^r_j}}\sum_{k'=\bar{b}^r_j+1}^{U^r_j}\left(\sum_{i\in\mathcal{I}}M^r_{i,j,k'}\bar{w}^r_{i,j}\right)y^r_{j,k'}+\sum_{\substack{j\in\mathcal{J}'\\ \bar{b}^r_j>L^r_j}}\sum_{k'=L^r_j+1}^{\bar{b}^r_j}\left(\sum_{i\in\mathcal{I}}m^r_{i,j,k'}\bar{w}^r_{i,j}\right)(1-y^r_{j,k'})+\bar{\varepsilon}^r\leq 0,\quad(35)$$

where $\bar{b}^r_j=\bar{b}_{J-j}$, $y^r_{j,k'}=y_{J-j,k'}$, $U^r_j=U_{J-j}$, $L^r_j=L_{J-j}$, $\bar{w}^r_{i,j}=\bar{w}_{I+\bar{b}_{J-j}+1-i,J-j}$, as in Table 1. With an inverted transformation, it is equivalent to:

$$-\sum_{\substack{j\in\mathcal{J}'\\ \bar{b}_j<U_j}}\sum_{k'=\bar{b}_j+1}^{U_j}\left(\sum_{i\in\mathcal{I}}M^r_{I+\bar{b}_j+1-i,J-j,k'}\bar{w}_{i,j}\right)y_{j,k'}$$
$$+\sum_{\substack{j\in\mathcal{J}'\\ \bar{b}_j>L_j}}\sum_{k'=L_j+1}^{\bar{b}_j}\left(\sum_{i\in\mathcal{I}}m^r_{I+\bar{b}_j+1-i,J-j,k'}\bar{w}_{i,j}\right)(1-y_{j,k'})+\bar{\varepsilon}\leq 0.\quad(36)$$

Notice that, since $M_{i,j,k}\neq M^r_{N+1-i+k,m-j,k}$ and $m_{i,j,k}\neq m^r_{N+1-i+k,m-j,k}$, big-$M$ and small-m parameters are not transformed back. Thus, the reversed cut (36) and the original cut (32) differ from each other. The reversed cuts can also be tightened as in Proposition 3.1.

3.4 Combinatorial cut

Besides the original and the reversed cut, a combinatorial cut (Codato and Fischetti, 2006) can also be generated, and it is as follows:

$$\sum_{j\in\mathcal{J}':\ \bar{b}_j<U_j}y_{j,\bar{b}_j+1}\geq 1.\quad(37)$$

The combinatorial cut implies that a feasible allocation requires an increase in the capacity of at least one buffer. This cut is valid due to the monotonicity: the sub-problem will be tighter than or equivalent that of the simulated infeasible solution if the capacity of no buffer is increased, thus, ε is not decreasing, and there is no possibility to obtain a feasible solution. This combinatorial cut is also applied in Weiss and Stolletz (2015).

3.5 Cut selection

According to Section 3.1, 3.3 and 3.4, each time an infeasible buffer solution is simulated, three cuts are generated (original, reversed, and combinatorial). Including all of them in the master problem can define a tighter formulation, but it also increases the solution complexity. Thus, selecting which cut(s) to add to the master problem in each iteration can have a relevant impact on the algorithm performance.

The conditions under which the combinatorial cut is tighter than the original are defined in the following (the proof is in Appendix I). It can also be applied to the reversed cut.

Proposition 3.2. *Combinatorial cut* (37) *is tighter than cut* (33) *if* $k_j = \bar{b}_j + 1$ *for all* $j \in \mathcal{J}'$ *such that* $\bar{b}_j < U_j$.

If the combinatorial cut is tighter than both the original and the reversed cuts, only the combinatorial cut is added to the master problem. If the combinatorial cut is tighter than only one of the two cuts, only the cut tighter than the combinatorial is added. Otherwise, if the combinatorial is looser than the other two cuts, it is discarded. The other two cuts may make similar partition, and adding both of them may bring high computational burden in each iteration instead of reducing the number iterations. By setting a user-defined coefficient c, the original cut and the reversed cut can be considered similar if, for all j:

$$\left|\sum_{i\in\mathcal{I}} M^r_{I+\bar{b}_j+1-i,J-j,k'}\bar{w}_{i,j} - \sum_{i\in\mathcal{I}} M_{i,j,k'}\bar{w}_{i,j}\right| \le c\frac{\bar{\varepsilon}}{J-1},\ \forall \bar{b}_j + 1 \le k' \le U_j$$

$$\left|\sum_{i\in\mathcal{I}} m^r_{I+\bar{b}_j+1-i,J-j,k'}\bar{w}_{i,j} - \sum_{i\in\mathcal{I}} m_{i,j,k'}\bar{w}_{i,j}\right| \le c\frac{\bar{\varepsilon}}{J-1},\ \forall L_j + 1 \le k' \le \bar{b}_j$$

If the two cuts are similar, one is randomly added to the master problem, otherwise both of them are added. The larger the value of c, the higher the probability to include only one cut.

The algorithm follows the classic Benders scheme as follows. The buffer spaces b_j are initially set to the lower bound L_j. An iterative procedure will be repeated by simulation, cut generation, cut selection, updating and solving the master problem and updating the buffer solution. The procedure stops when a feasible solution is visited.

4 Numerical analysis

Numerical tests have been performed to evaluate the efficiency of the proposed approach and analyze the property of buffer allocation in various production systems. A first experiment compares the computation time with a state-of-the-art sample-path exact algorithm through a variety of system characteristics. A second experiment investigates the performance of the proposed approach when the stage number increases and assess its problem size limit. Then, the property of buffer allocation of some typical system settings and several long realistic flow lines with unreliable machines are studied. A comparison with a state-of-the-art approximate method is also conducted, and the results are shown in Appendix III of the online supplemental material for reasons of conciseness.

The algorithms are implemented in Java and Cplex 12.10 is used for solving the master problem. The experiments are conducted on a cluster of DELL M630 with Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz processors and 256 Gb RAM. Cplex can use up to 16 threads. The time limit for solving each instance is set to 5 hours, i.e., 18000 seconds.

4.1 Computation time comparison: proposed algorithm vs. state-of-the-art algorithm

The first experiment addresses the BAP in systems with various characteristics. The lower and upper bounds (L_j, U_j) of the buffer capacity are set to 1 and 20, respectively. The number of jobs I is set to $1,000,000$. Such value has been chosen to avoid sampling errors in the processing time probability distributions. For each instance of the experiment, this number of jobs assures to reach the steady state value of the throughput evaluated through simulation. All the machines have Lognormal distributed processing time. There are two types of machines in the system, bottleneck and non-bottleneck, where the bottleneck machines (BN) have longer average processing times than non-bottleneck. For BN machines, the Lognormal distribution has the mean of the normally distributed logarithm μ equal to 2, while for non-bottleneck machines it is equal to 1.5.

Various factors can be relevant to the complexity of BAP. The number of machines J relates to the number of variables and constraints. More buffer spaces will be needed if higher target throughput or/and higher processing time variability are set, hence larger infeasible region, leading to more computational effort to confirm the lower bound. Another factor that could be relevant is the position of bottlenecks, since it affects the throughput. However, the mechanism of how the bottleneck position will affect BAP is not trivial, so the effect is studied through numerical analysis. Four relevant factors are identified: machine numbers, target throughput, processing time variability and bottleneck position. The target throughput is implicitly controlled as *target efficiency* η^*, which is the ratio between the system throughput and the throughput of the bottleneck in isolation. The processing time variability is controlled through the coefficient of variation, denoted by $CV = \mu/\sigma$. Systems with zero (balanced), one and two bottlenecks are all studied.

The benchmark algorithm is proposed by Weiss and Stolletz (2015), who applied Benders decomposition and combinatorial cuts (37) to the formulation proposed in Matta (2008). They also proposed a decomposition framework that first decomposes the system into sub-systems and solves BAP of each sub-system to accelerate the procedure. This decomposition is not included in the benchmark algorithm, since the proposed algorithm can be also embedded into the framework as the algorithm to solve the BAP of sub-systems. The aim of this experiment is to show that the proposed model is tighter than that proposed in Matta (2008), and, consequently, the Benders cuts generated from the proposed method can be more efficient than the combinatorial cuts.

4.1.1 Full factorial experiment of two-bottleneck systems

A full factorial design of two-bottleneck systems is applied. Table 2 shows the experimental design. The machine number is set to 4 and 6. Two levels of target efficiency are used (0.66, 0.72), and two levels of CV are investigated (0.85, 0.9). Six levels of bottleneck positions are considered to represent the cases of both BN at the beginning of the line (FF), both at the end (LL), both in the middle (MM), and mixed cases (FL, FM, ML). Table 2b shows the machine indexes of BNs, both for the 4-machine and the 6-machine lines.

For each of the 48 combinations of factors, five replicates are run. In total, 240 instances are solved with both the proposed and the state-of-the-art algorithms.

In the experiment, the proposed approach (algorithm $a2$) always solves the problem within 3500 seconds, while the state-of-the-art approach ($a0$) solves only 37 out of 48 cases within one hour. Figure 3 reports, on a logarithmic scale, the interval plots of the

Factor	**Notation**	**Levels**
Stage number	J	$4, 6$
Target efficiency	η^*	$0.66, 0.72$
Processing time variability	CV	$0.85, 0.9$
BN positions		as in Table 2b

(a) Design of experiment

	Notation					
J	FF	LL	MM	FL	FM	ML
4	1,2	3,4	2,3	1,4	1,3	2,4
6	1,2	5,6	3,4	1,6	1,4	3,6

(b) Bottleneck positions

Table 2: Full factorial experiment of two-bottleneck systems

ratio between the computation time of the state-of-the-art approach ($T0$) and that of the proposed approach ($T2$). The limits of the intervals are calculated assuming normality of data and using quantiles of normal distribution with confidence interval equal to 95%. The same approach is used for all interval plots of the paper. The cases where the state-of-the-art approach runs out of time are represented by empty labels and $T0$ set to 5 hours, otherwise by solid labels. The figure shows that the proposed approach is faster than the state-of-the-art in many cases. Such cases are represented by dots above the horizontal dash line in the figure. Also, in several cases the proposed approach can be orders of magnitude faster. $T0$ is smaller than $T2$ in a few exceptions, most of them with low target efficiency. In these cases, $T2$ values are up to 165 seconds, meaning that both the algorithms could solve these instances in a very short time. With high target efficiency, the state-of-the-art-approach is faster in only three cases, all with FL bottleneck position; however, the difference of the computation time between the two algorithms is up to 400 seconds only.

Besides the total computation time, the optimization and simulation times are also separately analyzed. The optimization time is the total amount of time for solving the master problems with Cplex. Figure 4a shows the interval plots of the ratio between the optimization time of the state-of-the-art approach ($CplexT0$) and that of the proposed

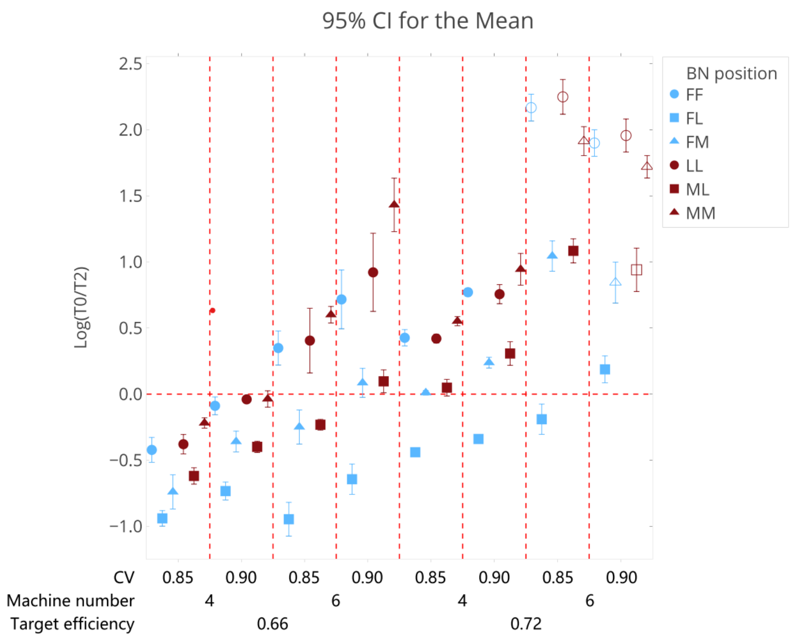
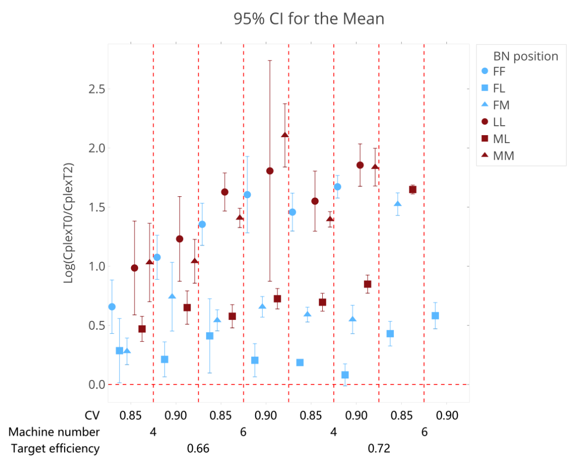


Figure 3: Total computation time comparison: proposed approach ($T2$) with state-of-the-art ($T0$). Five replicates are run for each combination of factors.

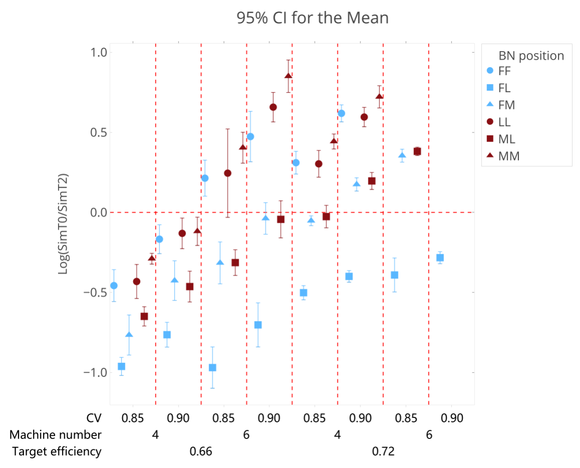
approach ($CplexT2$), while Figure 4b shows the same interval plots for the simulation time. In both figures, the cases where the state-of-the-art approach runs out of time are not shown. The proposed approach uses a shorter optimization time in almost all the cases, while the state-of-the-art approach uses shorter simulation time in most of the cases. It should be noticed that to generate combinatorial cuts only the simulation is needed, while for Benders cuts also the calculation of dual solutions is made. This takes more time and increases the simulation time of $a2$. For the easy cases (i.e., low values of η^*, J and CV), a few iterations are needed and simulation time dominates optimization time, and, hence, the state-of-the-art approach performs better. On the contrary, when optimization time dominates simulation time, the proposed approach is faster. Interestingly, the proposed approach improves the optimization time more significantly when bottlenecks are close to each other. The effect of bottleneck position is further discussed in Section 4.1.3.

4.1.2 One-bottleneck systems and balanced systems

In this section, six 6-machine lines with one bottleneck and four balanced lines are analyzed. The first five columns of Table 3 show the system identifiers and settings. Systems 1 to



(a) Optimization time comparison



(b) Simulation time comparison

Figure 4: Optimization and simulation time comparison: proposed ($T2$) and state-of-the-art approach ($T0$). Five replicates are run for each combination of factors.

6 are six-stage one-BN, with three levels of BN position, two levels of η^* and fixed CV. Systems 7 to 10 are balanced lines, one is 4-stage and three are 6-stage.

The last three columns of Table 3 show the results. Columns $T2$ and $T0$ report the average computation time of five independent replicates of the proposed and the state-of-the-art approaches, respectively. The optimal solutions are in column $\sum_{j\in\mathcal{J}} b_j$ (which is the same for all replicates). The proposed approach is able to solve all the cases within the 5 hour time limit, while the state-of-the-art cannot solve four cases (i.e., 4, 8, 9, 10). For three cases (1, 5, 7), which are the 4-machine balanced line, and two one-bottleneck 6-machine lines with BN at the first or last machine, the two approaches perform similarly. These cases are solved by both approaches in less than 5 minutes. For the other three cases (2, 3, 6), the proposed approach is one order of magnitude faster than the state-of-the-art.

4.1.3 Discussion

The results show that the proposed method outperforms the state-of-the-art for complex cases, but not for easy cases, as it saves optimization time at the cost of simulation time.

The effect of bottleneck position in the 2-BN experiment deserves attention. Figure 5

Case	J	BN-position	η^*	CV	$T2$	$T0$	$\sum_{j\in\mathcal{J}} b_j$
1	6	F	0.9	0.9	185.08	211.27	12
2	6	F	0.92	0.9	302.85	1243.26	14
3	6	M	0.9	0.9	410.82	6199.12	17
4	6	M	0.92	0.9	782.29	> 18000	21
5	6	L	0.9	0.9	180.28	298.56	12
6	6	L	0.92	0.9	337.26	1610.44	15
7	4	-	0.66	0.85	52.92	119.21	18
8	6	-	0.52	0.85	2641.13	> 18000	19
9	6	-	0.54	0.85	4391.11	> 18000	22
10	6	-	0.56	0.85	15726.78	> 18000	26

Table 3: Average computation time and optimal solution of one-bottleneck lines and balanced lines. Five replicates are run for each combination of factors.

shows the computation time of the proposed approach, with the x-axis indicating the total buffer capacity. FL, FM, ML 6-machine systems require a smaller total buffer capacity, but the computation time is significantly larger than the other cases. Thus, besides the machine number and the total buffer capacity, which represent the size of the optimization model and the cardinality of the infeasible area, the position of BN machines is a third factor impacting the complexity of BAP. An intuitive reason is that, for distant bottleneck flow lines, there exist many local optima / sub-optima, which make searching for the global exact solution more difficult.

4.2 Computation performance on line length

In this section, the maximum system length that can be solved within the 5-hour time limit is studied. As BN position and the total buffer capacity are shown to be relevant factors impacting on the computation time, they are varied. To vary the total buffer capacity (which is an output of the experiment) the target efficiency has been varied. Two levels of η^* (0.66 and 0.72) and three levels of BN positions (MM, ML and LL) are analyzed. Other experiment settings are: processing times follow a Lognormal distribution with μ

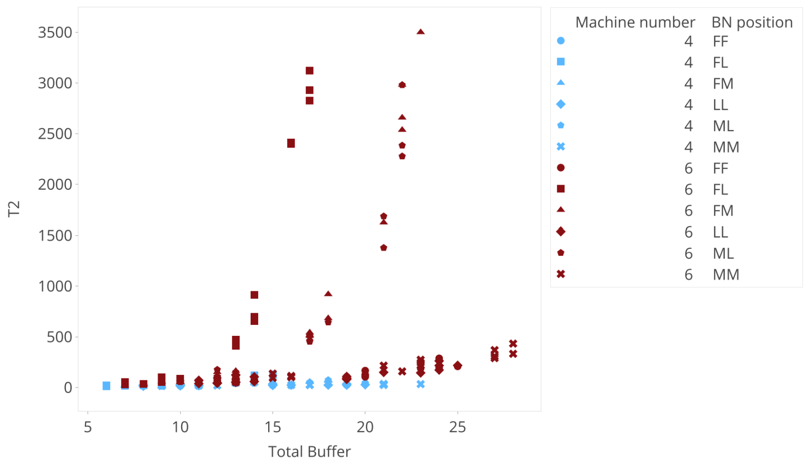


Figure 5: Computation time of the proposed approach. Five replicates are run for each combination of factors.

equal to 1.5 for non-BN machines, and equal to 2 for BN machines; CV is equal to 0.85. Only lines with an even machine number are studied, and for system with $2n$ machines, bottlenecks are located at the n-th and $(n+1)$-th machines for MM systems, n-th and $2n$-th machines for ML systems, and at the $(2n-1)$-th and $2n$-th for LL systems. Table 4 shows the maximum machine number (Max J), computation time ($T2$), total buffer capacity ($\sum_{j\in\mathcal{J}} b_j$) and buffer allocation (b_j) in each experiment setting. The table shows the results of one replicate for each line as illustrative example. It can be observed that the values of Max J are quite different among the various settings, and the maximum machine number, equal to 30, can be observed in LL system with target efficiency equal to 0.66.

4.3 System property study

The proposed approach has been applied to various 6-machine flow lines, both reliable and unreliable, to to study the property of buffer allocations. Reliable lines includes identical (namely balanced), V-shaped, inverted V-shaped and increasing/decreasing processing times. The processing times of all machines follow Lognormal distributions with CV equal to 0.85, while μ differs from one machine to the other. Unreliable lines includes identical (balanced), increasing/decreasing, and step-shaped mean time to repair (MTTR). Processing times of all machines are deterministic and equal to 1, and machine efficiency is equal to 0.8. Both time to failure and time to repair follow exponential distributions. MTTR is

BN-position	η^*	Max J	$T2$	$\sum_{j\in\mathcal{J}} b_j$	b_j
MM	0.66	14	11624.1	24	$1, 1, 1, 1, 1, 2, 9, 3, 1, 1, 1, 1, 1$
MM	0.72	8	4242.3	27	$1, 1, 4, 15, 3, 2, 1$
ML	0.66	20	5921.6	24	$1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1,$
					$1, 1, 2, 2$
ML	0.72	8	8705.1	21	$1, 1, 1, 4, 6, 3, 5$
LL	0.66	30	7613.79	37	$1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,$
					$1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 7$
LL	0.72	10	2495.21	25	$1, 1, 1, 1, 1, 1, 1, 4, 14$

Table 4: Performance on dimension with different experiment settings. Results of one replicate are shown as example.

varied from one machine to the other, and as the machine efficiency is fixed, the mean time to failure (MTTF) is changed accordingly.

Each system is first simulated with each buffer equal to 4, and the throughput is set as TH^* of the BAP. The simulation length is equal to 1,000,000 jobs. The bounds L_J and U_j of each buffer are equal to 1 and 20, respectively. For each system, five replicates are run.

In Table 5, the first column shows machine parameters (μ for reliable, MTTR for unreliable lines). The second column shows the system name, and the third reports the optimal total buffer capacity. The last column shows the detailed buffer allocations, where multiple optimal solutions are sometimes found in different replicates.

The well-known *storage bowl phenomenon* (Hillier et al., 1993) suggests that buffers closer to the center of the line should be equipped with larger capacity, but this phenomenon is not always found in the results. In the balanced reliable line or balanced unreliable lines with MTTR equal to 4, some optimal solutions show a storage bowl phenomenon, but they do not improve the total buffer capacity, compared with a balanced buffer allocation. For balanced unreliable lines with MTTR equal to 40 and 100, allocating more space to the central buffers and less in the marginal ones helps reducing the total buffer capacity by two and three slots, respectively (if compared with reference buffers). Thus, lines with less frequent but more time consuming failures show significant bowl phenomenon.

In the reliable unbalanced lines (i.e., *decreasing*, *increasing* and *V-shaped*) more space is allocated to the bottleneck machines. The *inverted V-shaped* reliable line shows a combination of bowl phenomenon and bottleneck effect, where extreme buffers have the largest capacity and the central buffer capacity is larger than the second and fourth buffer.

In unreliable lines with increasing and decreasing MTTR (i.e., *Decreasing*, *Increasing*, *Step 1*, *Step 2*), machines with large MTTR are separated by small buffers. The reason could be that once a failure occurs on those machines, the repair time is so long that only giant buffer space could significantly mitigate blockage and starvation. As failures are less frequent, the buffer is often not used. Thus, it is not beneficial to allocate large space here.

4.4 Solution of long realistic lines

The proposed approach has been tested on realistic lines. The cases studied in this section are all taken from the literature. Weiss and Stolletz (2015) were able to solve two 24-machine lines, which are the longest lines solved so far with exact methods. Moreover, Tempelmeier (2003) summarized some practical lines. In this experiment, the lines considered in Weiss and Stolletz (2015) and four systems in Tempelmeier (2003) have been addressed. In the following, the results are presented first for the 24-machine line of Weiss and Stolletz (2015) and then for the four lines of Tempelmeier (2003). These lines are solved with the proposed solution procedure ($a2$) and with the proposed procedure embedded in the system decomposition framework of Weiss and Stolletz (2015) ($a2$ *with decomposition*).

4.4.1 Long line with unreliable machines of Weiss and Stolletz (2015)

The 24-machine line of Weiss and Stolletz (2015) is composed of 22 reliable machines with deterministic or Erlang distributed processing times, and two unreliable machines with deterministic processing times and exponential TTR and TTF. The two unreliable machines are the bottlenecks of the line, and they are placed in the middle of the line (machines 12 and 13). The system is solved with a target throughput set to 90% of the bottleneck rate, and with $250,000$ jobs flowing in the system. Two cases, differing from each other by MTTF and MTTR, are studied. The first configuration has $MTTF = 5$, $MTTR = 2$, the second $MTTF = 10$, $MTTR = 4$. All the above mentioned settings are the same as in Weiss and Stolletz (2015). A five-hour time limit is set for both $a2$ and $a2$ *with decomposition*.

Reliable lines: processing time Log-norm(μ, $\sigma = 0.85\mu$)			
μ	System type	Total buffer	Buffer allocation
1.5, 1.5, 1.5, 1.5, 1.5, 1.5	Balanced	20	(4,4,4,4,4)(1,6,4,6,3)
4, 3.5, 5, 2.5, 2, 1.5	Decreasing	10	(6,1,1,1,1)
1.5, 2, 2.5, 3, 3.5, 4	Increasing	10	(1,1,1,1,6)
1.5, 2, 2.5, 2.5, 2, 1.5	V-shaped	11	(1,1,7,1,1)
2.5, 2, 1.5, 1.5, 2, 2.5	Inverted V-shaped	19	(7,1,3,1,7)
Unreliable lines: processing time = 1, machine efficiency = 0.8			
MTTR	System type	Total buffer	Buffer allocation
4, 4, 4, 4, 4, 4	Balanced 1	20	(2,4,5,7,2)(4,4,4,4,4)(3,4,5,4,4)(3,4,4,5,4)(3,5,5,4,3)
40, 40, 40, 40, 40, 40	Balanced 2	18	(2,3,6,6,1)(1,3,8,5,1)(1,6,7,3,1)(1,4,4,8,1)(1,2,12,1,2)
100, 100, 100, 100, 100, 100	Balanced 3	17	(1,48,3,1)(1,1,11,1,3)(1,2,11,2,1)(1,5,7,3,1)
60, 50, 40, 30, 20, 10	Decreasing	18	(1,1,6,5,5)(1,1,6,6,4)(1,1,2,13,1)(1,1,2,10,4)(1,2,4,8,3)
10, 20, 30, 40, 50, 60	Increasing	18	(3,3,10,1,1)(2,7,7,1,1)(3,10,3,1,1)(5,5,6,1,1)(5,4,7,1,1)
10, 10, 10, 30, 30, 30	Step 1	18	(1,7,5,4,1)(2,5,7,3,1)(2,7,5,3,1)(2,7,7,1,1)(1,6,7,3,1)(2,6,7,2,1)
60, 60, 60, 30, 30, 30	Step 2	18	(1,2,1,13,1)(1,4,5,6,2)(1,1,13,2,1)(1,4,1,11,1)(1,2,8,4,3)

Table 5: Results for system property study. Five replicates are run for each combination of factors.

Results can be found in Table 6. The first two columns report the values of MTTF and MTTR, while column $\sum_{j\in\mathcal{J}} b_j^*$ shows the optima of Weiss and Stolletz (2015). Columns 4-5 show the results of $a2$, while columns 6-7 of $a2$ *with decomposition*. For each algorithm, the mean computation time over the five replicates (column Time (s)) and the solution found at the end of the algorithm (column $\sum_{j\in\mathcal{J}} b_j$) are shown. If the value of $\sum_{j\in\mathcal{J}} b_j$ is followed by *(opt)*, the approach could find the optimum within the time limit, otherwise *(LB)* indicates that the solution is the lower bound found in the time limit.

The $a2$ approach takes on average 48 minutes to solve the line with $MTTF = 5, MTTR = 2$, but it cannot solve the second configuration within the time limit with an optimality gap equal to 4. With $a2$ *with decomposition*, the first configuration is solved in 127 minutes on average, thus the state-of-the-art decomposition framework slows down the algorithm. As for the second configuration the optimal solution cannot be found, but the optimality gap is reduced to 1 or 2, depending on replicates. From Weiss and Stolletz (2015), the state-of-the-art method (combinatorial cuts and the decomposition framework) is able to solve the line with $MTTF = 10, MTTR = 4$ in 52460 seconds on average. However, within 5-hour time limit, on the same computer, the state-of-the-art algorithm finds only a lower bound equal to 15, i.e., with optimality gap equal to 2.

			$a2$		$a2$ *with decomposition*	
MTTF	MTTR	$\sum_{j\in\mathcal{J}} b_j^*$	Time (s)	$\sum_{j\in\mathcal{J}} b_j$	Time (s)	$\sum_{j\in\mathcal{J}} b_j$
5	2	9	2890	9 (opt)	7628	9 (opt)
10	4	17	> 18000	13 (LB)	> 18000	15,16 (LB)

Table 6: Mean computation time and solutions of the 24-machine line. Five replicates are run for each system setting.

4.4.2 Long lines with unreliable machines of Tempelmeier (2003)

Tempelmeier (2003) introduced four real lines of different size and with a large total buffer capacity, denoted by line A, B, C, and D. Machines of the lines are unreliable, with deterministic processing times and exponential failure (whose values are the same as in Tempelmeier (2003)). The main characteristics of the four lines are shown in the first four columns of

Table 7. For each line, the table shows the number of stages (J), the target throughput (TH^*), which is the throughput achieved by the original system, and the total capacity of the real system indicated in Tempelmeier (2003) ($\sum_{j\in\mathcal{J}} b_{j,0}$). This total capacity is not optimal. For both tested algorithms ($a2$ and $a2$ with decomposition), one replicate is run for each line, with a simulation length equal to $500,000$ jobs, which guarantees that the simulation reaches the steady state. Columns 5-7 of Table 7 shows the results of $a2$, and columns 8-10 of $a2$ with decomposition. The final solution is simulated to get the throughput, denoted as TH_{all}.

Line D can be solved to optimality by $a2$ with decomposition in 9986 seconds. Instead, $a2$ itself cannot solve the problem within the time limit. In this case, embedding the solution procedure in the decomposition framework speeds up the optimization.

Lines A, B, and C could not be solved within five hours by either of the two procedures, and only the lower bound is provided. For instance, the buffer allocation found for line A by $a2$ has a total capacity equal to 30, and it can achieve $TH = 1.5597$, which is 9% less than the target throughput. Also $a2$ with decomposition is only able to find a lower bound, and its total capacity is equal to 27. While in line A the decomposition framework proposed by Weiss and Stolletz (2015) finds, in five hours, a worse lower bound, in lines B it actually finds a better lower bound. This suggests further studies on the combination of the proposed model and the decomposition framework.

Characteristics				$a2$			$a2$ with decomposition		
Line	J	TH^*	$\sum_{j\in\mathcal{J}} b_{j,0}$	Time (s)	$\sum_{j\in\mathcal{J}} b_j$	TH_{all}	Time (s)	$\sum_{j\in\mathcal{J}} b_j$	TH_{all}
A	19	1.7168	285	> 18000	30 (LB)	1.5597	> 18000	27 (LB)	1.5524
B	23	0.13390	459	> 18000	99 (LB)	0.1298	> 18000	156 (LB)	0.1307
C	8	0.003661	131	> 18000	82 (LB)	0.003634	> 18000	82 (LB)	0.003630
D	14	0.02694	166	> 18000	31 (LB)	0.02693	9986	44 (opt)	0.02694

Table 7: Results of four lines of Tempelmeier (2003). One replicate is solved for each line.

4.5 Comparison with a state-of-the-art meta-heuristic

To assess the differences between the proposed and an approximate approach, a comparison with a state-of-the-art meta-heuristic is conducted. Among the few works (Weiss et al., 2019) solving the primal BAP (i.e., minimizing the total buffer capacity subject to a throughput constraint), the parallel tabu search by Costa et al. (2015) is used as benchmark.

The comparison addresses small, medium and large systems. System configurations, experiment settings and detailed results can be found in Appendix III. The meta-heuristic by Costa et al. (2015) can almost reach optimality in medium-small lines; for realistic and complex lines, the exact method finds solutions that save a relevant amount of buffer spaces.

5 Conclusion

This work proposes a new formulation for the sample-path-based BAP. Several techniques are developed to speed up the Benders-decomposition-based exact solution approach, which include the simulation-based cut generation approach, the generation of Benders cut of the reversed system, the tighter valid inequality of the Benders cuts, the combinatorial cut, and a cut selection heuristic. Numerical comparisons show that the proposed model and solution approach can be orders of magnitude faster than the state-of-the-art approach in several cases. A 24-machine line from Weiss and Stolletz (2015) can be solved to optimality in reasonable time. Among the experiments of this work, the longest solved line has 30 machines and optimal total buffer capacity equal to 37, with a computation time equal to 7614 seconds. Embedding the proposed approach into the state-of-the-art decomposition framework of Weiss and Stolletz (2015) enables to solve a real 14-machine line proposed by Tempelmeier (2003), which the proposed approach itself cannot solve. The optimum is equal to 44 and computation time is 9986 seconds. However, this decomposition may sometimes worsen the algorithm performance, depending on system characteristics. Future study will be dedicated to improve the decomposition framework, so that more complex cases can be solved within reasonable time.

Numerical analyses also show that the computation time is affected by several factors. Besides machine number, processing time variance and target throughput, which are already known to be relevant, the position of bottlenecks also influences the computation time. When bottlenecks are distant from each other, the BAP becomes harder to solve. The reason could be the existence of multiple local optima, which will be further investigated. For distant bottleneck flow lines, an efficient solution generation algorithm should be developed in the future to improve the procedure.

In this work, the master problem of Benders decomposition is solved to optimality with a generic MILP solver, which is the computational bottleneck of the approach. Customized solution generation procedures should be able to improve the efficiency. Last, the proposed approach can be extended to the BAP of assembly / disassembly / closed-loop systems.

The source code supporting the results is openly available at https://github.com/myrazhang/BAP_Int.

References

Andijani, A. (1998). A multi-criterion approach for kanban allocations. *Omega 26*(4), 483–493.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik 4*(1), 238–252.

Codato, G. and M. Fischetti (2006). Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research 54*(4), 756–766.

Conway, R., W. Maxwell, J. O. McClain, and L. J. Thomas (1988). The role of work-in-process inventory in serial production lines. *Operations research 36*(2), 229–241.

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.

Enginarlar, E., J. Li, S. M. Meerkov, and R. Q. Zhang (2002). Buffer capacity for accommodating machine downtime in serial production lines. *International Journal of Production Research 40*(3), 601–624.

Gershwin, S. (1994). *Manufacturing Systems Engineering.* Prentice Hall.

Hillier, F. S., K. C. So, and R. W. Boling (1993). Toward characterizing the optimal allocation of storage space in production line systems with variable processing times. *Management Science 39*(1), 126–133.

Inman, R. R. (1999). Empirical evaluation of exponential and independence assumptions in queueing models of manufacturing systems. *Production and Operations Management 8*(4), 409–432.

Li, J. (2013). Continuous improvement at toyota manufacturing plant: applications of production systems engineering methods. *International Journal of Production Research 51*(23-24), 7235–7249.

Li, J., D. E. Blumenfeld, N. Huang, and J. M. Alden (2009). Throughput analysis of production systems: recent advances and future topics. *International Journal of Production Research 47*(14), 3823–3851.

Matta, A. (2008). Simulation optimization with mathematical programming representation of discrete event systems. In *2008 Winter Simulation Conference*, pp. 1393–1400. IEEE.

Matta, A., G. Pedrielli, and A. Alfieri (2014). Event relationship graph lite: Event based modeling for simulation-optimization of control policies in discrete event systems. In *Proceedings of the Winter Simulation Conference 2014*, pp. 3983–3994. IEEE.

Nori, V. S. and B. R. Sarker (1998). Optimum number of kanbans between two adjacent stations. *Production Planning & Control 9*(1), 60–65.

Pedrielli, G., A. Matta, A. Alfieri, and M. Zhang (2018). Design and control of manufacturing systems: a discrete event optimisation methodology. *International Journal of Production Research 56*(1-2), 543–564.

Powell, S. G. and D. F. Pyke (1996). Allocation of buffers to serial production lines with bottlenecks. *IIE transactions 28*(1), 18–29.

Seo, D.-W., S.-S. Ko, and U. Jung (2009). Optimal buffer allocation in tandem queues with communication blocking. *ETRI Journal 31*(1), 86–88.

Seo, D.-W. and H. Lee (2011). Stationary waiting times in m-node tandem queues with production blocking. *IEEE Transactions on Automatic Control 56*(4), 958–961.

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

Tempelmeier, H. (2003). Practical considerations in the optimization of flow production systems. *International Journal of Production Research 41*(1), 149–170.

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

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

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

Yamazaki, G., T. Kawashima, and H. Sakasegawa (1985). Reversibility of tandem blocking queueing systems. *Management Science 31*(1), 78–83.

Zhang, M. and A. Matta (2020). Models and algorithms for throughput improvement problem of serial production lines via downtime reduction. *IISE Transactions 52*(11), 1189–1203.