OR Spectrum (2015) 37:869–902
DOI 10.1007/s00291-015-0393-z

REGULAR ARTICLE

Buffer allocation in stochastic flow lines via sample-based optimization with initial bounds

Sophie Weiss · Raik Stolletz

Received: 4 June 2014 / Accepted: 16 February 2015 / Published online: 13 March 2015

Abstract The allocation of buffer space in flow lines with stochastic processing times is an important decision, as buffer capacities influence the performance of these lines. The objective of this problem is to minimize the overall number of buffer spaces achieving at least one given goal production rate. We optimally solve this problem with a mixed-integer programming approach by sampling the effective processing times. To obtain robust results, large sample sizes are required. These incur large models and long computation times using standard solvers. This paper presents a Benders Decomposition approach in combination with initial bounds and different feasibility cuts for the Buffer Allocation Problem, which provides exact solutions while reducing the computation times substantially. Numerical experiments are carried out to demonstrate the performance and the flexibility of the proposed approaches. The numerical study reveals that the algorithm is capable to solve long lines with reliable and unreliable machines, including arbitrary distributions as well as correlations of processing times.

Keywords Buffer allocation · Stochastic flow lines · Benders Decomposition · Sampling · Bounds

1 Introduction

Flow lines consist of a number of stations that are arranged in series and separated by limited buffer spaces. The workpieces flow through the system from station to station, waiting in the buffer if the downstream station is not available. This type of

S. Weiss (✉) · R. Stolletz
Business School, Chair of Production Management, University of Mannheim,
Schloss, 68131 Mannheim, Germany
e-mail: weiss@bwl.uni-mannheim.de

production system is often applied in practice, mainly for mass production. Examples can be found in the automotive industry (Colledani et al. 2010; Li 2013) and in food production (Cooke et al. 2005; Liberopoulos and Tsarouhas 2005), among others.

Burman et al. (1998) note that there is a great potential in the systematic optimization of the buffer allocation in such stochastic flow lines, as it highly influences the performance of the line. The stochastic influences are due to random machine breakdowns, uncertain times of repair, and random processing times. This can lead to blocking and starvation of the stations, which may lead to a reduction of the throughput. The allocation of additional buffer space may increase the throughput, although it leads to an increase of the average work in process in the line. In this paper, we develop an optimization approach for the buffer allocation in a linear flow line with all those stochastic influences.

Two basic streams of research can be found regarding the allocation of buffers in stochastic flow lines: performance evaluation and optimization. Dallery and Gershwin (1992) and Gershwin and Schor (2000) provide an overview of the different evaluation approaches. *Exact evaluation* is only possible for small lines as analytical results are difficult to obtain (Li and Meerkov 2009). For longer lines, simulation and other approximative methods, e.g., decomposition or aggregation, are applied. The methods proposed in the literature on performance evaluation can also be used as integral parts of optimization approaches by applying generative methods and evaluative methods iteratively. The generative methods are used to obtain candidate solutions that are then evaluated. The optimization of buffer allocations, referred to as Buffer Allocation Problem (BAP) in literature, is NP-hard (MacGregor Smith and Cruz 2005). Three types of objective functions can be found: minimization of the total buffer capacities with respect to a given goal throughput, throughput maximization with respect to a limited number of buffers, and profit maximization. This paper focuses on the minimization of total buffer capacities.

The optimization approaches can be divided into exact approaches, heuristics, and rules of thumb. Demir et al. (2014) provide an overview on the approaches published after 1998. *Exact approaches* only exist for small lines because of the combinatorial complexity and the lack of exact evaluation methods (MacGregor Smith and Cruz 2005; Li and Meerkov 2009). Recently, sample-based approaches have been proposed to optimize flow lines with limited buffer capacities. For sufficiently large sample sizes, the obtained allocations converge to the exact solution. Matta and Chefson (2005) propose an iterative change of configurations to determine buffer allocations based on a mathematical programming formulation developed by Schruben (2000) and Chan and Schruben (2008). Matta (2008) presents an exact mixed-integer programming (MIP) formulation that optimizes the number of buffer spaces behind each station, using samples of the processing times in continuous time.

Heuristic methods based on samples are developed by Gürkan (2000), Helber et al. (2011) and Alfieri and Matta (2012, 2013). Gürkan (2000) uses sample-based gradient estimates of performance measures to obtain buffer allocations in continuous lines. She points out that this approach may also be used to approximate lines with discrete goods. Helber et al. (2011) present a discrete-time linear programming (LP) formulation that incorporates the BAP. The authors use sampling to transform the stochastic processing times of the different jobs at a given sta-

tion into the corresponding realizations of production capacities per discrete time period. This method leads to simulation and discretization errors. Alfieri and Matta (2012) introduce the concept of time buffers, which can be used to derive approximate buffer allocations. This approach can also be applied to reduce the feasible region of the buffer capacities as necessary in Matta (2008). Recently, Alfieri and Matta (2013) proposed a time-based decomposition approach that solves the mathematical programming formulation by iteratively solving a number of subsystems. These subsystems contain only a portion of the entities in the whole model. The subsystems are connected via additional constraints reflecting the status of the system defined by previous subsystems. Other heuristic methods include Tabu Search and Simulated Annealing, as generative methods, in combination with simulation or decomposition, as evaluation methods (Lutz et al. 1998; Spinellis and Papadopoulos 2000). Yamashita and Altiok (1998) and Diamantidis and Papadopoulos (2004) apply Dynamic Programming in combination with decomposition or aggregation. In addition to the risk of obtaining local optima as final solutions, some of these methods are based on restrictive assumptions. Caramanis (1987) applies Generalized Benders Decomposition with gradient estimates for performance approximation. However, due to errors in the gradient estimates, optimal solutions cannot be guaranteed. Li and Meerkov (2009) propose heuristics based on closed formulas and recursion approaches. They show that these heuristics are fast, but do not necessarily provide good allocations.

Rules of thumb based on extensive numerical studies are proposed by Hillier et al. (1993), Powell and Pyke (1996), and Hillier (2000). However, these results may not be generalized, and a large computational effort is needed for their derivation.

This paper deals with exact sample-based MIP formulations, i.e., the obtained results are sample-exact. The advantage of these sampling approaches compared to other approaches proposed in literature is based on their flexibility: besides the ability to cope with both reliable and unreliable lines, they do not require the assumption of statistical independency. The processing times, times to failure, and repair times can follow any distribution, or may be taken from empirical data. However, when using standard solvers, the sample-based MIP formulations proposed in literature remain intractable for flow lines with more than three stations due to extensive computation times (Matta 2008). Therefore, to exploit the flexibility of these approaches, a fast solution method has to be developed. We develop a Benders Decomposition approach for such a MIP formulation of the BAP.

The main contribution of this paper is to develop a Benders Decomposition approach with combinatorial cuts to optimally and efficiently solve the BAP with respect to an underlying sample. The performance of this algorithm is improved via the derivation of initial bounds. The numerical study shows the great degree of flexibility of this approach, as its sample-based structure allows to take account for correlations and arbitrary distributions of processing times, times to failure, and repair times.

This paper is organized as follows. Section 2 introduces the MIP formulation for the optimization of flow lines. In Sect. 3, the Benders Decomposition approach and a procedure to obtain initial bounds are presented. Section 4 provides a numerical study on the performance of Benders Decomposition and the initial bounds. Section 5 presents the conclusions and further research efforts.

2 Sample-based flow line model

This section formulates the evaluation problem and the optimization problem with respect to the buffer allocation in flow lines. First, the underlying assumptions are given in Sect. 2.1. The Benders Decomposition approach is based on iterative generation of candidate allocations and evaluation of these candidates. Therefore, Sect. 2.2 presents a fast simulation algorithm for throughput evaluation for a given buffer allocation. Finally, Sect. 2.3 describes the mixed-integer program for buffer optimization.

The key idea of the sample-based modeling approach is to simulate the flow of a large number of workpieces throughout the line. Therefore, the start and departure times of processing a workpiece, w, at a station, s, are represented by a set of real-valued decision variables. The random processing times are replaced by a deterministic sample. The samples are generated by descriptive sampling (DS) (Saliby 1990a). In DS, deterministic values serve as the input for the inverse distribution function. These values are then shuffled randomly to represent random behavior. This method is more appropriate than simple random sampling (SRS) because it leads to a more precise description of the underlying distribution (Saliby 1990b). Moreover, DS leads to a reduction of the variability of the input sample and therefore to a reduction of the variability of the simulation results. The numerical study in Sect. 4.1 supports this claim in the case of the BAP [see also Stolletz and Weiss (2013)].

The samples consist of effective processing times, i.e., the repair times are assumed to be included in the (raw) processing times. This can be accomplished with a single distribution or the sum of the distributions of processing times and repair times.

2.1 Assumptions

The model of the flow line is based on the following assumptions:

- The flow line consists of S stations, which process W workpieces.
- A number of W_0 workpieces corresponds to the warm-up phase.
- The maximum capacity of the buffer behind station s is limited to B_s.
- The material supply to the first station is unlimited, i.e., the first station never starves.
- The buffer behind the last station is infinitely large. Thus, this station cannot be blocked.
- The processing times of the workpieces at each station are generally distributed or deterministic. The MIP uses sampled processing times, $d_{s,w}$, for each station, s, and each workpiece, w.
- The stations may be subject to operation-dependent failures. Times to failure and repair times are generally distributed. Sampled repair times are added to the sampled processing times, $d_{s,w}$, of the workpiece w, which is processed when the breakdown of station s occurs.
- In the event of blocking, the station finishes the currently processed workpiece. Then, the workpiece waits at the station until a buffer space or the following station becomes available (blocking after service).
- Transportation times are insignificant or are already included in the processing times.

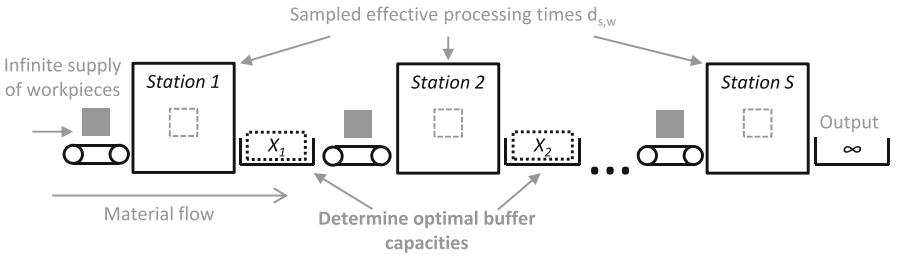


Fig. 1 Flow line under consideration

– A minimum throughput rate of TH^* has to be reached after the warm-up.

Figure 1 shows an example of a flow line according to these assumptions.

2.2 Evaluation of given allocations

If the capacities of the buffers are known, the start times and the departure times of each workpiece at each station can be derived using a fast simulation algorithm, as Algorithm 1. The corresponding notation can be found in Table 1.

$XS_{1,1} = 0$
for all stations $s < S$ **do**
  $XF_{s,1} = XS_{s,1} + d_{s,1}$
  $XS_{s+1,1} = XF_{s,1}$
end for
$XF_{S,1} = XS_{S,1} + d_{S,1}$
for all workpieces $w > 1$ **do**
  for all stations $s < S$ **do**
    if $s = 1$ **then**
      $XS_{s,w} = XF_{s,w-1}$
    else
      $XS_{s,w} = \max\{XF_{s,w-1}, XF_{s-1,w}\}$
    end if
    if $X_s = 0$ **then**
      $XF_{s,w} = \max\{XS_{s,w} + d_{s,w}, XF_{s+1,w-1}\}$
    else if $X_s \geq w$ **then**
      $XF_{s,w} = XS_{s,w} + d_{s,w}$
    else
      $XF_{s,w} = \max\{XS_{s,w} + d_{s,w}, XS_{s+1,w-X_s}\}$
    end if
  end for
  $XS_{S,w} = \max\{XF_{S,w-1}, XF_{S-1,w}\}$
  $XF_{S,w} = XS_{S,w} + d_{S,w}$
end for

Algorithm 1: Throughput calculation

Table 1 Notation for the models

Indices	
$w = 1, \ldots, W$	Workpieces
$s = 1, \ldots, S$	Stations in the flow line
$b = 0, \ldots, B_s$	Possible buffer capacities behind station s
Parameters	
$d_{s,w}$	Processing time of workpiece w at station s
TH*	Minimum throughput
W_0	Number of workpieces in the warm-up phase
M	BigM (sufficiently large positive number)
Real-valued decision variables	
$\mathrm{XS}_{s,w}$	Start time of workpiece w at station s
$\mathrm{XF}_{s,w}$	Departure time of workpiece w from station s
X_s	Buffer capacity behind station s
Binary decision variables	
$Y_{s,b} = \begin{cases} 1 & \text{If the buffer behind station } s \text{ is equal} \\ 0 & \text{Otherwise} \end{cases}$	

The algorithm calculates the start and departure times of each workpiece w at each station s. The first workpiece starts processing at the first station at time zero (line 1) and flows through the line without ever being blocked, because the line is empty. Consequently, it leaves a station s after the processing time elapsed (line 3) and starts processing at the subsequent station $s + 1$ as soon as it leaves s (line 4). Lines 7–24 model the flow of the remaining workpieces. Start times $\mathrm{XS}_{s,w}$ of stations $s = 2, \ldots, S$ depend on the availability of the workpiece w. Since the first station never starves, processing of a workpiece w starts when $w - 1$ leaves the station (line 10). At stations $s = 2, \ldots, S$, it may happen that no workpieces are available. In this case, s idles until station $s - 1$ provides a workpiece (lines 12 and 22). Departure times $\mathrm{XF}_{s,w}$ of stations $s = 2, \ldots, S - 1$ depend on the downstream buffer capacities X_s and the occurrence of blocking. If the capacities X_s are set to zero, a workpiece w leaves station s after it finished processing and the subsequent station becomes available (that is, workpiece $w - 1$ leaves station $s + 1$, line 15). In contrast, if buffer spaces are allocated behind station s, but the available buffer capacity suffices for all workpieces in the system, blocking can never occur (line 17). If there are more workpieces in the system than buffer capacities at station s, blocking may occur. Therefore, workpiece w leaves station s when its processing is completed and a buffer space becomes available (that is, a workpiece leaves the buffer, because it starts processing at station $s + 1$, line 19). The last station S is never blocked and consequently, workpieces leave this station directly after processing (line 23).

Based on this information, the realized throughput TH is calculated by the fraction of the number of finished parts $W - W_0$ and the required time $\mathrm{XF}_{S,W} - \mathrm{XF}_{S,W_0}$ after the warm-up phase:

$$\mathrm{TH} = \frac{W - W_0}{\mathrm{XF}_{S,W} - \mathrm{XF}_{S,W_0}} \tag{1}$$

2.3 Optimization of buffer allocations

The problem of allocating a minimum number of total buffer spaces while achieving a given minimum throughput can be solved by a MIP formulation as follows. Additionally to the notation used in Sect. 2.2, a binary variable $Y_{s,b}$ is required to indicate that the buffer capacity behind station s equals b.

$$\text{Minimize} \sum_{s=1}^{S-1} X_s \tag{2}$$

$$\text{s.t.} \quad \mathrm{XS}_{s,w} + d_{s,w} \leq \mathrm{XF}_{s,w}, \qquad \forall s,\ \forall w \tag{3}$$

$$\mathrm{XF}_{s,w} \leq \mathrm{XS}_{s+1,w}, \qquad \forall s \leq S-1,\ \forall w \tag{4}$$

$$\mathrm{XF}_{s,w} \leq \mathrm{XS}_{s,w+1}, \qquad \forall s,\ \forall w \leq W-1 \tag{5}$$

$$\mathrm{XF}_{S,W} - \mathrm{XF}_{S,W_0} \leq \frac{W - W_0}{\mathrm{TH}^*}, \tag{6}$$

$$\mathrm{XS}_{s+1,w} - \mathrm{XF}_{s,w+b} \leq M \cdot (1 - Y_{s,b}), \qquad \forall s \leq S-1,\ \forall b,\ \forall w \leq W-b \tag{7}$$

$$\sum_{b=0}^{B_s} Y_{s,b} = 1 \qquad \forall s \leq S-1, \tag{8}$$

$$X_s = \sum_{b=0}^{B_s} b \cdot Y_{s,b} \qquad \forall s \leq S-1, \tag{9}$$

$$\mathrm{XS}_{s,w}, \mathrm{XF}_{s,w} \geq 0, \qquad \forall s,\ \forall w \tag{10}$$

$$Y_{s,b} \in \{0, 1\} \qquad \forall s \leq S-1,\ \forall b \tag{11}$$

The objective function (2) minimizes the overall number of buffer spaces in the line. The constraints are linearizations of the formulas given in Algorithm 1. Constraint (3) states that a workpiece, w, departs from station s at the earliest time after being processed. Consequently, the slack of the inequality corresponds to the blocking time of workpiece w after being processed at station s. A workpiece cannot start being processed by station $s+1$ until it departs from station s. This is ensured by the inequality described by (4). The slack of this inequality defines the waiting time of workpiece w in the buffer between station s and station $s+1$. As a station can only process one workpiece at a given time, the inequality in (5) states that workpiece $w+1$ does not start processing at station s until the preceding workpiece w departs from this station. A station may starve between the processing of two consecutive workpieces, which is equivalent to the slack of Constraint (5). Inequality (6) ensures that a minimum throughput, TH*, is reached [see Equality (1)]. Constraint (7) states that the buffer capacity is not exceeded. If $b = X_s$, the inequality ensures that workpiece

w departs from the buffer between stations s and $s+1$ before workpiece $w+b$ enters. Otherwise, the inequality is deactivated by the BigM on the right-hand side (RHS). We choose BigM as the product of the maximum possible buffer capacity, $\max_s B_s$, and the maximum processing time, $\max_{s,w} d_{s,w}$. If there is no buffer between station s and station $s+1$, i.e., $b=0$, the inequality reduces to $\mathrm{XS}_{s+1,w} \leq \mathrm{XF}_{s,w}$. Together with Inequality (4), the departure time of workpiece w at station s is assured to equal the starting time of w at station $s+1$. Compared to the formulation presented by Matta (2008), we assume blocking after service instead of blocking before service. The capacity of each buffer between two stations must be unique. This is stated in Eq. (8). Constraint (9) connects the (redundant) buffer space variables X_s and the binary variables $Y_{s,b}$. Variables X_s are used for notational convenience.

Note that the combination of Equalities (4) and (5) determines the start times as in Algorithm 1. Accordingly, the combination of Eqs. (3) and (7) determines the completion times.

If the buffer capacities behind each station are given, the MIP can also be used for evaluation (instead of Algorithm 1). However, the throughput may be overestimated, because the warm-up phase is based on the number of workpieces instead of a specific point in time. This results in a degree of freedom regarding the start and departure times in the warm-up phase of the optimal solution. Due to this flexibility, the workpieces do not necessarily start processing as soon as possible. To avoid this overestimation, the start and departure times have to be added to the objective function (2).

3 Application of Benders Decomposition to the Buffer Allocation Problem

The complexity of the MIP presented in the previous section incurs long computation times. Therefore, it is necessary to apply certain techniques to reduce the computation time. One literature stream concerns decomposition methods, which aim to split the original problem into smaller parts and to solve them iteratively. One of these methods is Benders Decomposition (Benders 1962). Benders Decomposition divides the original problem into a master problem and a subproblem, both of which are solved iteratively. The master problem is a relaxation of the original problem, calculates a solution, and passes it to the subproblem. The subproblem uses this solution to generate cuts that contain information about the feasibility and optimality of the current master solution. These cuts are added to the master problem such that optimality is proven at the termination of the algorithm. Consequently, a sequence of master- and subproblems has to be solved to obtain an optimal solution of the original problem.

3.1 Adjustments and specific features

Figure 2 provides an overview of the decomposition procedure for the BAP. The master problem contains only binary and integer decision variables. The subproblem considers the remaining variables, assuming that the variables of the master problem are fixed. In the case of the MIP formulation presented in Sect. 2, the binary variables, $Y_{s,b}$, and the integer variables, X_s, become part of the master prob-

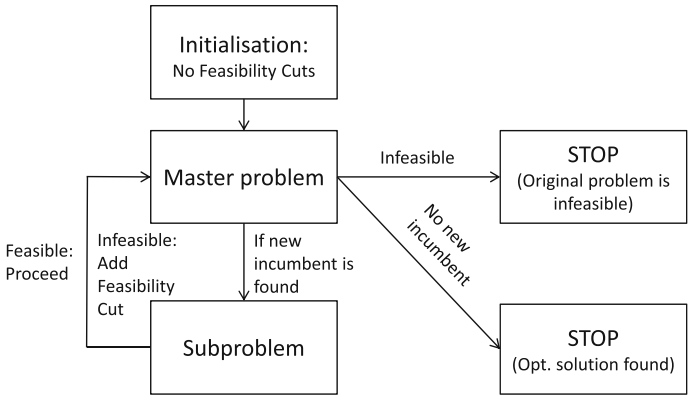


Fig. 2 Overview of Benders Decomposition for buffer allocation

lem. The real-valued decision variables, $\mathrm{XS}_{s,w}$ and $\mathrm{XF}_{s,w}$, belong to the subproblem.

Constraints (3)–(6) and (10) only contain real-valued decision variables and thus belong to the subproblem. Constraints (8)–(11) are included in the master problem, as they only contain binary variables. Constraint (7) contains both types of variables. It forms part of the subproblem and contains the master variables, X_s, as parameters. Consequently, Constraint (7) can be replaced by (12).

$$\mathrm{XS}_{s+1,w} - \mathrm{XF}_{s,w+X_s} \leq 0, \quad \forall s \leq S-1, \ \forall w \leq W - X_s \tag{12}$$

Moreover, as the integer variables are assumed to be known when the subproblem is solved, the subproblem reduces to the evaluation version of the MIP. Note that the objective function (2) includes no real-valued decision variables. Thus, the objective function of the master problem is equal to the objective function (2). To avoid an overestimation of the throughput as outlined in Sect. 2.3, we use Algorithm 1 to evaluate the throughput of a given buffer allocation. The feasibility of this throughput is then checked by comparison to the goal throughput, TH^*.

The information on feasibility is expressed in additional constraints, which include only the integer variables. We add these constraints, called feasibility cuts, to the master problem. If the master problem contains all of the feasibility cuts, it is equivalent to the original problem.

In general, an exponential number of such constraints exists, which are usually not known in advance. Therefore, we consider a relaxed master problem, which includes no feasibility constraints at the beginning of the solution process. By iterating between the relaxed master problem and the subproblem, additional cuts are generated to ensure the feasibility of the final solution. If the subproblem is feasible, the resulting solution is optimal.

Based on Eqs. (2) and (8)–(11), the complete master problem is defined as follows.

$$\text{Minimize} \sum_{s=1}^{S-1} X_s \tag{2}$$

$$\text{s.t.} \sum_{b=0}^{B_s} Y_{s,b} = 1 \quad \forall s \leq S-1, \tag{8}$$

$$X_s = \sum_{b=0}^{B_s} b \cdot Y_{s,b} \quad \forall s \leq S-1, \tag{9}$$

Feasibility cuts

$$Y_{s,b} \in \{0,1\} \quad \forall s \leq S-1, \ \forall b \tag{11}$$

If the master problem is infeasible, the original problem is also infeasible because the master problem is a relaxation of the original problem as long as not all feasibility cuts are added. Because of the restriction of the buffer capacities to B_s, unboundedness cannot occur in the master problem. The subproblem cannot be unbounded because it is a simple evaluation. If the original problem has an optimal solution, the algorithm finishes after a finite number of iterations when the subproblem does not return new feasibility cuts.

As described in the literature on Benders Decomposition, the feasibility cuts are obtained from inequality (13) (classical feasibility cut).

$$0 \geq -\left(\sum_{s=1}^{S-1} \sum_{w=1}^{W-b_s} \mu^h_{5,s,w,b_s} \cdot M \cdot (1 - Y_{s,b_s}) + \mu^h_4 \cdot \frac{W - W_0}{\text{TH}^*} - \sum_{s=1}^{S} \sum_{w=1}^{W} \mu^h_{1,s,w} \cdot d_{s,w} \right) \tag{13}$$

μ^h is an extreme ray. The cut only contains the binary variables associated with the buffer capacities in the current solution. Note that we use the LP to solve the subproblem in the case of the classical feasibility cut, as information from the dual subproblem is needed for the extreme rays. Because the original formulation uses BigM coefficients in Constraints (7), the classical feasibility cuts (13) are weak. As a solution, Codato and Fischetti (2006) propose combinatorial cuts for Benders Decomposition. These cuts force at least one variable to be changed and exclude the redundant constraints that are caused by the usage of BigM coefficients. For the BAP, more information is available. We develop new Combinatorial Cuts based on the following observations. If the current buffer allocation is infeasible, the capacity of at least one buffer has to be increased. If the buffer capacities are decreased, the throughput remains the same or decreases and the goal throughput cannot be reached. Therefore, all solutions that include only the combinations of smaller buffer capacities than the current solution are known to be infeasible as well. We propose the following combinatorial cut if the current buffer capacity behind station s equals b_s:

$$1 \leq \sum_{s=1}^{S-1} \sum_{b=b_s+1}^{B_s} Y_{s,b}. \tag{14}$$

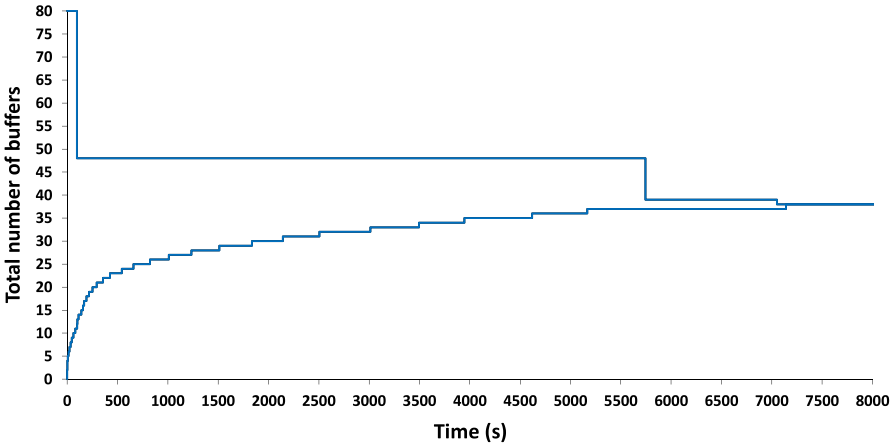


Fig. 3 Course of the lower and upper bounds during the solution process

The RHS sums all the variables of possible buffer capacities for every station that are larger than the current buffer capacities ($b > b_s$). At least one of these variables must assume a value of one, i.e., at least one of the buffers increases.

3.2 Generation of lower bounds from subsystems

Figure 3 depicts the solution process using Benders Decomposition with combinatorial cuts for an exemplary flow line with 5 stations, a sample size of $W = 250{,}000$ workpieces, and a bottleneck at the end of the line. One can observe that the solver takes only a few steps to find upper bounds that are close to the optimum, while the lower bound increases in many small steps. This is because if a candidate solution reaches the goal throughput, the optimal total buffer capacity has to be smaller or equal to the total buffer capacity of this solution. In contrast, if a candidate solution does not fulfill the requirement of the goal throughput, it does not necessarily mean that the total buffer capacity of this solution has to be increased. There may be other solutions with the same total number of buffer spaces (or even less) but with a different allocation that are feasible. Therefore, it is crucial to find appropriate lower bounds.

In literature, numerous studies propose algorithms, which approximate the optimal buffer allocation. Li and Meerkov (2009) propose several approaches to approximate the optimal solution for lines with more than three stations, which have deterministic processing times and stochastic up and down times of the stations. To use these solutions as bounds, they have to be evaluated. Depending on whether the solution is feasible or infeasible, it serves as a feasibility cut or as an upper bound on the total number of buffer spaces. Derivation of guaranteed lower bounds or individual upper bounds cannot be accomplished with these approaches. Therefore, we focus on the generation of guaranteed lower bounds and compare the different strategies in the numerical study in Sect. 4.

We decompose the line into several subsystems assuming that the supply of the first station of each subsystem is unlimited. As a result, the effect of starvation, which can occur in the original line, is neglected for the first station in each subsystem. Additionally, it is assumed that the workpieces can always leave the subsystem. Therefore, the last station of each subsystem is never blocked. Thus, for given buffer capacities, the isolated subsystem will never have a lower throughput than the original system as proven in the following theorems.

Theorem 1 *In steady-state, the throughput of a system with unlimited supply at the first station is higher or equal to the throughput of an identical system with limited supply.*

Proof Let $\text{Arr}_w \geq 0$ be the arrival time of workpiece w in the system with limited supply. According to Algorithm 1, the start times of workpieces 1 and 2 at the first station of the system with limited supply are calculated from $\text{XS}^{\text{lim}}_{1,1} = \text{Arr}_w$ and $\text{XS}^{\text{lim}}_{1,2} = \max\{\text{XF}_{1,1}, \text{Arr}_w\}$, respectively. As $\text{Arr}_w = 0$ for all w in the system with unlimited supply, the start times equal $\text{XS}^{\text{unl}}_{1,1} = 0$ and $\text{XS}^{\text{unl}}_{1,2} = \text{XF}_{1,1}$. Consequently, $\text{XS}^{\text{unl}}_{1,2} \leq \text{XS}^{\text{lim}}_{1,2}$. With mathematical induction using the above formulas for w and the formulas of Algorithm 1 to calculate start and departure times, it follows that $\text{XF}^{\text{unl}}_{S,W} \leq \text{XF}^{\text{lim}}_{S,W}$, i.e., less time to produce W workpieces is required in the unlimited case, and therefore, the throughput of the system with unlimited outflow is higher or equal to the throughput of an identical system with limited outflow. □

Theorem 2 *In steady-state, the throughput of a system with unlimited outflow at the last station is higher or equal to the throughput of an identical system with limited outflow.*

Proof Let $\text{Dep}_w \geq 0$ be the time, workpiece w is allowed to leave the system with limited supply. According to Algorithm 1, the departure time of workpiece 1 at the last station, S, is calculated as $\text{XF}^{\text{lim}}_{S,1} = \max\left\{\text{XS}_{S,1} + d_{S,1}, \text{Dep}_1\right\}$ for the system with limited outflow. As $Dep_w = 0$ for all w in the system with unlimited outflow, the departure time equals $\text{XF}^{\text{unl}}_{S,1} = \text{XS}_{S,1} + d_{S,1}$. Consequently, $\text{XF}^{\text{unl}}_{S,1} \leq \text{XF}^{\text{lim}}_{S,1}$. With mathematical induction using the above formulas for w and the formulas of Algorithm 1 to calculate start and departure times, it follows that $\text{XF}^{\text{unl}}_{S,W} \leq \text{XF}^{\text{lim}}_{S,W}$, i.e., less time to produce W workpieces is required in the unlimited case, and therefore, the throughput of the system with unlimited outflow is higher or equal to the throughput of an identical system with limited outflow. □

Consequently, the optimal buffer capacities of the subsystems are lower than or equal to the optimal buffer capacities in the original line. Levantesi et al. (2001) use lower bounds from subsystems of size 2 as a starting point for a gradient algorithm to approximate optimal buffer allocations in continuous lines.

The larger the subsystems are, the better the original setting is approximated. However, for large subsystems, the computation time may be long. Therefore, we propose an iterative procedure. We first solve subsystems with $i = 2$ stations, as shown in Fig. 4. Each solution of a subsystem provides a certain buffer capacity that forms a lower bound for the respective buffer. These buffer capacities are then used as lower

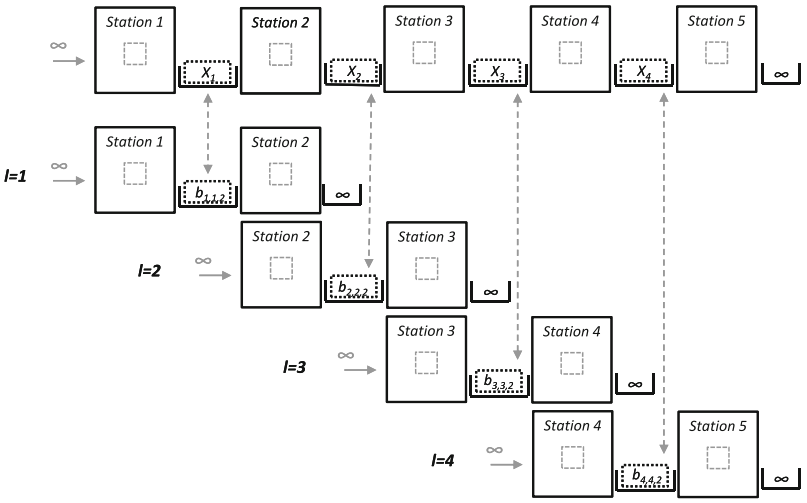


Fig. 4 Generation of lower bounds via subsystems of size $i = 2$

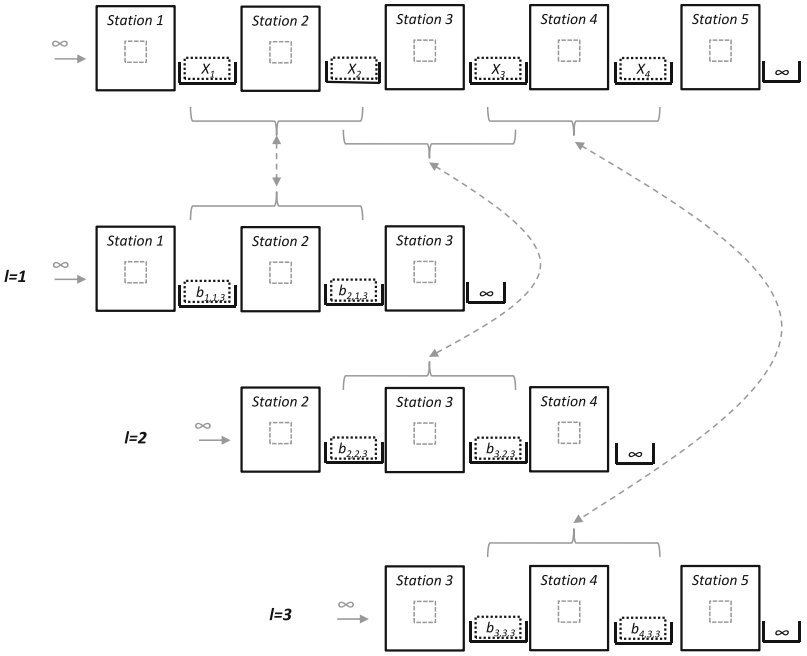


Fig. 5 Generation of lower bounds via subsystems of size $i = 3$

bounds in the original system and all of the subsequent subsystems. In the next step, we solve the subsystems of size $i \geq 3$. The optimal buffer capacity of each subsystem $l = 1, \ldots, S - i + 1$ of size i at station s is denoted by $b_{s,l,i}$. Figures 4 and 5 depict a line with 5 stations divided into subsystems of size $i = 2$ and $i = 3$, respectively.

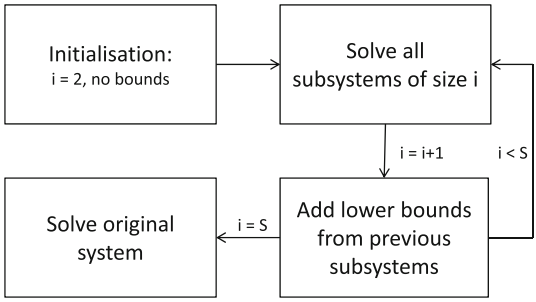


Fig. 6 Overview of bound calculation

In contrast to the subsystems of size $i = 2$, the lower bounds derived from the subsystems of larger sizes do not form bounds for individual buffers. Individual bounds, i.e., $b_{s,l,i} \leq X_s$ for $i \geq 3$, may force a certain buffer to be larger than necessary in the original line, resulting in a sub-optimal final solution for the original line. This is because the buffer allocation of the subsystem, which is found by the solver, may not be unique, as only the total number of buffer spaces is minimized. However, their sum forms a lower bound for all of the respective buffer capacities in the original line. Figure 5 illustrates this case for a subsystem of size $i = 3$. Inequality (15) presents the bounds obtained from subsystems of size i.

$$\sum_{j=0}^{i-2} b_{j+l,l,i} \leq \sum_{j=0}^{i-2} X_{j+l} \qquad \forall l \tag{15}$$

We apply Benders Decomposition to solve each subsystem. The size of the subsystems is increased iteratively, until the size of the original line is reached. This procedure is depicted in Fig. 6.

4 Numerical study

All of the algorithms are implemented in C++. Gurobi 5.0, with default settings, is used to solve the linear and mixed-integer programs. The numerical study is performed on an Intel Core i7-3930K with 6× 3.2 GHz and 32 GB RAM.

For all instances, the capacity of each buffer is limited to $B_s = 20$, and the warm-up phase is chosen to be $W_0 = 2000$.

To further speed up the solution process, we use callbacks, i.e., the master problem is not solved to optimality before handing over the values of the binary variables to the subproblem. Instead, a potential incumbent solution (the best integer solution found at any point of the search) is tested by the subproblem algorithm whenever the solver identifies one. If the solution is feasible, it becomes the new incumbent solution, and the solution process continues. Otherwise, a feasibility cut (13) or (14) is added to the master problem. We thereby avoid proving optimality in every step and visiting the nodes several times during different runs of the master problem. Both aspects waste

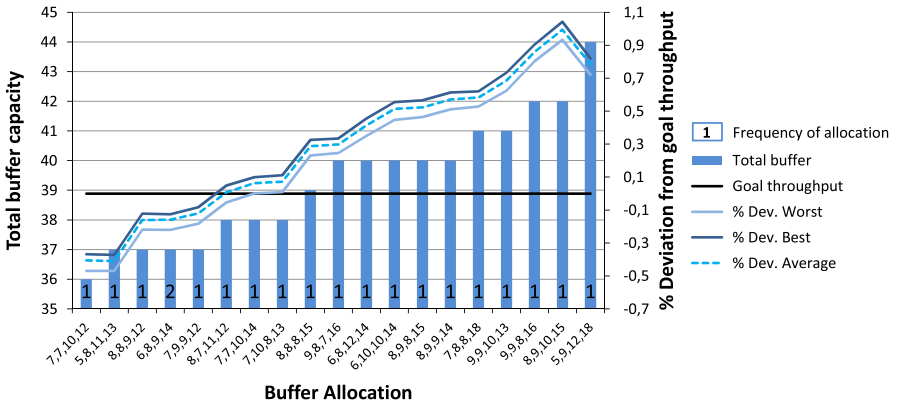
time (Bai and Rubin 2009). Note that an implementation without callbacks would lead to complete enumeration for the BAP.

4.1 A note on robustness

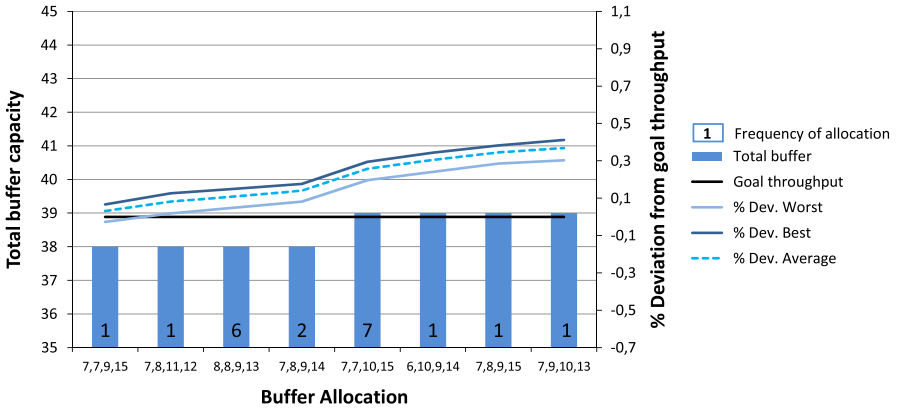
We investigate the robustness based on the instances from the numerical study of Matta (2008). We assume a line with 5 stations and a bottleneck at the end. The processing times are exponentially distributed, with a base processing rate of 7.0. The processing rate of the bottleneck is assumed to be 6.0. The goal throughput is set to 5.776.

Figure 7 depicts the results of a throughput evaluation for different optimal buffer allocations for a varying number of workpieces. These allocations are obtained by independently solving 20 samples with 10,000 (Fig. 7a), 250,000 (Fig. 7b), 1,000,000 (Fig. 7c), and 5,000,000 workpieces (Fig. 7d) each. The throughput evaluation is conducted with 20 additional samples of 5,000,000 workpieces. Figure 7 presents the relative deviation of the minimum, average, and maximum throughput from the goal throughput that is obtained by these 20 samples for each buffer allocation. For 10,000 workpieces (Fig. 7a), the independent optimization of 20 samples leads to 19 different buffer allocations. The total buffer capacity lies between 36 and 44 for the different samples. For a total number of buffer spaces of 39 or above, the goal throughput is always reached, whereas a total number of 37 (or less) is not (even in the best case) sufficient. On average, the goal throughput is reached for the allocations with a buffer capacity of 38 in total. This means that in the case of the allocation with 44 buffer spaces in total, 6 redundant buffer spaces (14 % of the total buffer space needed) are allocated in the line. In Fig. 7b (250,000 workpieces), only 8 different buffer allocations are obtained with a total number of 38 or 39 buffer spaces in the line. On average, the goal throughput is always reached for all allocations. Even in the worst case, the maximum deviation from the goal throughput equals 0.03 %. Figure 7c shows very similar results for $W = 1,000,000$, with a maximum deviation of 0.01 %. Therefore, it can be concluded that 250,000 workpieces is sufficient to obtain robust results for the given configuration. However, for increasing number of stations or increasing squared coefficients of variation (SCV), additional workpieces may be required to obtain robust results, because more different allocations are obtained (see Tables 11 and 12 in the Appendix, respectively). Figure 7d shows that the algorithm converges to a unique solution of the total buffer capacity, i.e., 38 buffer spaces are allocated. Two allocations result from the optimization of 20 samples, which both always reach the goal throughput.

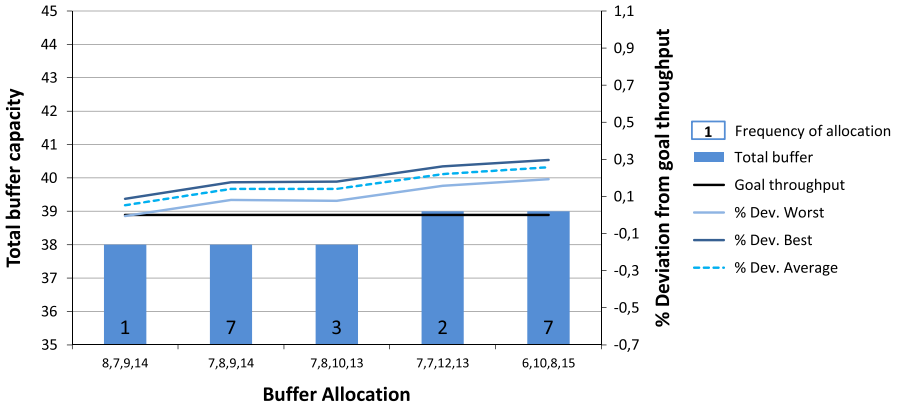
Figure 8 shows the results of a throughput evaluation for the different optimal buffer allocations obtained from samples generated with simple random sampling (SRS) instead of descriptive sampling (DS), as explained in Sect. 2 ($W = 250,000$). Compared to the results in Fig. 7b, the total number of buffer spaces varies between 37 and 39. The total number of different solutions obtained for 20 samples is 11 for SRS instead of 8 for DS. Moreover, the maximum deviation from the goal throughput is 0.03 % for DS. In contrast, in the case of SRS, a maximum deviation of 0.15 % is observed. Consequently, this demonstrates that DS leads to more robust results than SRS.



(a) $W = 10,000$

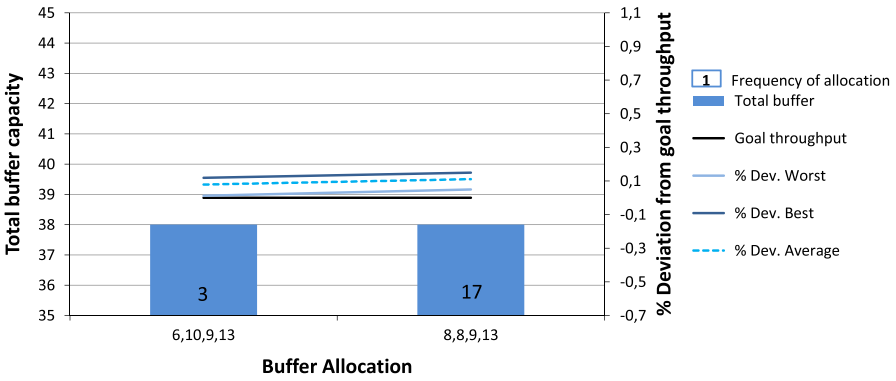


(b) $W = 250,000$



(c) $W = 1,000,000$

Fig. 7 Robustness of the approach regarding the no. of workpieces ($S = 5$, bottleneck last)



(d) $W = 5,000,000$

Fig. 7 continued

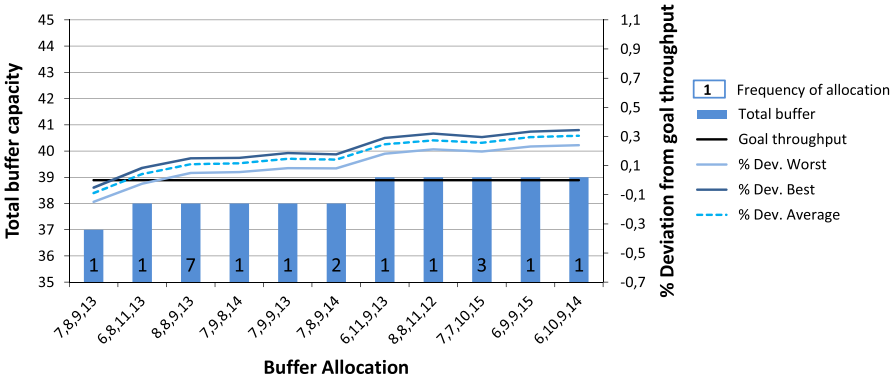


Fig. 8 Robustness of simple random sampling ($S = 5$, $W = 250{,}000$, bottleneck last)

4.2 Impact of bounds

This subsection compares three types of bounds: bounds derived from rules of thumb, bounds obtained from the optimal allocation (theoretical best case), and bounds generated from the subsystems as described in Sect. 3.2.

We use the rules of thumb developed by Powell and Pyke (1996) to generate allocations for given total buffer capacities. Powell and Pyke (1996) point out that balanced allocations lead to better throughput unless the imbalance caused by the bottleneck is more than 20 %. In the case of an imbalance of more than 20 %, the buffer capacity of the buffer, which is located farthest from the bottleneck, shall be decreased. The available buffer space shall be placed around the bottleneck. Infeasible allocations are used as feasibility cuts (14), while feasible allocations are upper bounds.

We investigate bounds from the optimal allocation (theoretical best case) to show the impact of near-optimal buffer allocations. In general, however, this solution is not known and can only be approximated, e.g., by rules of thumb and heuristics. The

Table 2 Time saving potential of approximate solutions

Type of bound	Feasible	Infeasible	Computation time (s)	Time savings (%)
None			7142	–
Rules of thumb				
8, 9, 9, 11		×	7214	−1
9, 9, 9, 10		×	7250	−2
8, 9, 10, 11		×	5753	20
9, 9, 10, 10		×	5834	18
Theoretical best cases				
7, 8, 9, 13		×	5721	20
8, 7, 9, 13		×	5860	18
8, 8, 8, 13		×	7093	1
8, 8, 9, 12		×	4892	32
8, 8, 9, 13	×		4711	34
Subsystems			69	99

optimal solution provides the best upper bound for the buffer capacities. Moreover, solutions that are infeasible but close to the optimum are good candidates for feasibility cuts. Therefore, as upper bound, the optimal solution is used, whereas $S-1$ feasibility cuts can be generated, each by decreasing the optimal capacity of a buffer by one.

The bounds generated from the subsystems according to Sect. 3.2 are of a different type as they provide (individual) lower bounds instead of only feasibility cuts.

Table 2 demonstrates the benefit of using different types of bounds for the exemplary flow line, which is described in the previous chapter. The first row shows the computation time without bounds of 7142 s as a reference. For the rules of thumb and the theoretical best cases, column 1 shows the tested allocations. Each of these allocations results either in a feasibility cut or a (non-individual) upper bound. We apply the rules of thumb for total buffer capacities of 37 and 38. Columns 2 and 3 depict whether the evaluation of the allocations results in a feasible or an infeasible throughput. The fourth and the fifth column show the computation times using these bounds and the resulting time savings in comparison to the calculation without bounds.

It can be observed that, in most of the cases, the bounds have a positive impact on the computation time. The feasibility cuts generated from rules of thumb with 38 buffer spaces in total reduce the computation time by around 20 %. In contrast, the feasibility cuts with a total buffer capacity of 37 have little impact on the computation time. The effect of feasibility cuts generated from the theoretical best cases varies for the different allocations from 1 to 32 %. The upper bound obtained from the optimal solution leads to the highest decrease in computation time (34 %). However, even in this case, the impact is rather low. Moreover, approximate solutions generated by rules of thumb or heuristics, in general, are worse than the allocations generated from known optimal solutions, which further reduces the usefulness of such bounds. In contrast, the (individual) lower bounds generated from the subsystems reduce the computation time by 99 %. Therefore, it is more advantageous to implement the lower

Table 3 Mean computation times (exponential)

S	Bottleneck	W	Original formulation	Computation time (s)			
				Benders Decomposition			
				Cl. Cut	Comb. Cut		
					Without callbacks	Without bounds	With bounds
3	Middle	10.000	306	8806	4	<1	<1
3	Last	10.000	906	6060	2	<1	<1
3	Middle	250.000	>10,000	>10,000	9	5	<1
3	Last	250.000	>10,000	>10,000	6	4	<1
5	Middle	10.000	>10,000	>10,000	>10,000	1745	1
5	Last	10.000	>10,000	>10,000	>10,000	2392	3
5	Middle	250.000	>10,000	>10,000	>10,000	5724	38
5	Last	250.000	>10,000	>10,000	>10,000	6720	66
7	Middle	10.000	>10,000	>10,000	>10,000	>10,000	1134
7	Last	10.000	>10,000	>10,000	>10,000	>10,000	5402
7	Middle	250.000	>10,000	>10,000	>10,000	>10,000	5998
7	Last	250.000	>10,000	>10,000	>10,000	>10,000	7484

bounds generated from the subsystems as described in Sect. 3.2 instead of feasibility cuts or upper bounds from near-optimal solutions. Due to this reason, we omit further investigations of these upper bounds and feasibility cuts and focus on the lower bounds obtained from the subsystems.

4.3 Exponentially distributed processing times

The investigation of instances with exponentially distributed processing times is based on the instances from the numerical study of Matta (2008), but varies the number of stations and the location of the bottleneck. The distribution of the processing times is as described in Sect. 4.1. We test instances with 3, 5, and 7 stations with bottleneck at the end of the line or in the middle of the line. We generate 10 independent samples for each configuration. As the original MIP formulation is able to solve only small instances, we use samples of 10,000 workpieces to demonstrate the improvements in the computation time of Benders Decomposition. However, Sect. 4.1 shows that this sample size is not sufficient to obtain robust results. Therefore, further studies use samples with $W = 250{,}000$.

Table 3 presents the computation times of complete enumeration, the original formulation, Benders Decomposition with classical feasibility cuts (Cl. Cut), and benders decomposition with combinatorial feasibility cuts (Comb. Cut). In the latter case, we present both results with and without initial bounds. The computation time is limited to 10,000 s. Only two settings are solvable within this time limit using the original MIP or the Benders Decomposition approach with classical cuts.

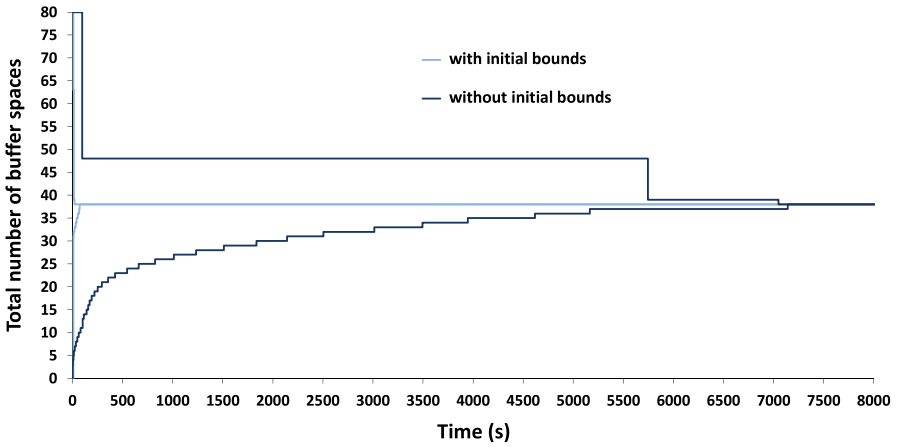


Fig. 9 Course of the lower and upper bounds during the solution process ($S = 5, W = 250{,}000$, bottleneck last)

Benders Decomposition with combinatorial cuts finds the optimal solution much faster than the implementation with classical cuts. This matches the findings of Codato and Fischetti (2006). Even the implementation without callbacks leads to faster computation times. However, callbacks are required to solve instances with more than 3 stations. The procedure with Combinatorial Cuts without bounds is able to solve instances with up to 5 stations within the time limit. The additional computation time of Benders Decomposition with Classical Cuts is composed of the computation time due to the usage of the LP and the computation time that stems from the weakness of the cut. Benders Decomposition with Combinatorial Cuts and initial bounds solves all instances to optimality within a reasonable amount of time.

Table 3 also shows that the instances with a bottleneck in the middle of the line are easier to solve than the instances with a bottleneck at the end. The reason is that a bottleneck in the middle of the line is covered by more subsystems. Therefore, the obtained bounds are better, which results in a smaller feasible region.

To analyze the impact of the initial bounds, Fig. 9 compares the course of the lower and upper bounds for Benders Decomposition with Combinatorial Cuts, with and without initial bounds, for one sample of a 5-station line with 250,000 workpieces and bottleneck at the end. To derive the lower bounds, we optimized four 2-station subsystems, three 3-station subsystems, and two 4-station subsystems. The computation of the bounds is completed after 8 s, with a lower bound of 31 buffer spaces for the whole line. The lower bound for the case without bounds slowly rises by 1 in each step. In the case with initial bounds, the optimal solution of 38 is found after 19 s and is proven after 69 s. Without bounds, the upper bound drops in large steps until the optimal solution is found after 7051 s. This solution is proven to be optimal after 7141 s.

Figure 10 depicts the shares of computation time for the bound calculation, the time until the optimal solution is found by the upper bound, and the time until this

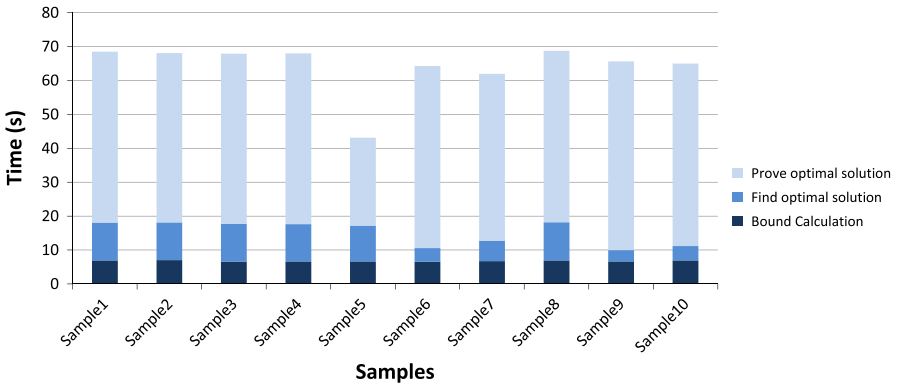


Fig. 10 Share of computation times for bound calculation and optimality proof ($S = 5$, $W = 250{,}000$, bottleneck last)

Table 4 Parameter settings for the base case

Number of stations S	7
Number of workpieces W	250,000
Distribution	Erlang-k
Squared coefficient of variation (SCV)	0.25
Base processing rate	0.5
Bottleneck	Middle
Processing rate of bottleneck	90 % of base rate
Goal throughput TH*	90 % of bottleneck rate

solution is proven to be optimal for a 5-station line with 250,000 workpieces and a bottleneck at the end. Most of the computation time is needed for the optimality proof. The calculation of the bounds represents only a small proportion of the total time, ranging from 9 to 15 % of the total computation time.

4.4 Generally distributed processing times

The following experiments give further insights on the performance of Benders Decomposition with Combinatorial Cuts and initial bounds. We investigate the performance of the algorithm with respect to generally distributed effective processing times. The generation of instances focuses on a base case, which is adapted from Helber et al. (2011) according to Table 4. We generate 10 independent samples for each configuration.

The experiment varies the distribution of the effective processing times and the number of stations based on the study in Helber et al. (2011). The Erlang-k distribution is used to generate processing times with squared coefficients of variation of 0.25 and 0.5, while the balanced mean variant of the Cox-2 distribution (Buzacott and Shanthikumar 1993) is used to generate processing times with squared coeffi-

Table 5 Mean computation times (Erlang-k and Cox-2)

Distribution	S	SCV	Bottleneck	Range of total buffer capacities	Computation time (s)		Max. deviation from goal throughput (%)
					Benders Decomposition (Comb. Cut)		
					Bounds	Total	
Erlang-4	5	0.25	Middle	6	<1	<1	0.36
Erlang-4	7	0.25	Middle	10	2	3	−0.05
Erlang-2	5	0.5	Middle	14	1	3	0.05
Erlang-2	7	0.5	Middle	22	27	76	−0.09
Cox-2	5	1.0	Middle	29–30	4	17	−0.22
Cox-2	7	1.0	Middle	46–47	308	1786	−0.04
Cox-2	5	2.0	Middle	60–62	20	74	−0.26
Cox-2	7	2.0	Middle	95–98	1509	6075	−0.36

Table 6 Detailed results (Erlang-k distribution, $S = 5$)

Sample	SCV	Optimal allocation	Max. dev. from TH* (%)	Initial bounds								
				$i = 2$				$i = 3$			$i = 4$	
				b_1	b_2	b_3	b_4	$\sum_{j=1}^{2} b_j$	$\sum_{j=2}^{3} b_j$	$\sum_{j=3}^{4} b_j$	$\sum_{j=1}^{3} b_j$	$\sum_{j=2}^{4} b_j$
1–10	0.25	1, 2, 2, 1	0.36	**1**	1	1	**1**	**3**	3	**3**	**5**	**5**
1–7, 9, 10	0.5	3, 4, 4, 3	0.52	1	2	2	1	6	6	6	10	10
8	0.5	3, 5, 3, 3	0.05	1	2	2	1	5	6	5	10	10

cients of variation 1.0 and 2.0, respectively. The number of stations is set to 5 and 7, respectively.

The computational results are given in Table 5. The first four columns describe the setting. Column 5 gives the range of the total number of buffer spaces in the optimal solutions of 10 samples. The average computation times for the bounds and the total time are given in columns 6 and 7. The last column presents the maximum deviation from the goal throughput of all samples.

The instances with low SCV are solved quickly. The reason is that the initial lower bounds are better for small SCVs, as less starving and blocking occurs; see Tables 6, 10, 11, and 12 in the Appendix. Tables 6, 10, 11, and 12 present the values of the optimal solutions and the initial bounds for all of the subsystems of all of the samples (samples with identical bounds and identical optimal solutions are aggregated in a single line). For an SCV of 0.25, some initial bounds are tight (marked in bold).

Instances with Cox-2 distributed processing times and 7 stations are especially difficult to solve, the computation time takes more than 1 h on average.

Table 5 also shows the computation time for the initial bounds. For Erlang-k instances with an SCV of 0.25, the calculation of the initial bounds takes a significant proportion of the total amount of computation time, summing up to approximately 50 % or even more. With increasing SCV, this proportion decreases. In the case of an SCV of 2.0, the portion of the bound calculation accounts for approximately 15 % and less of the total time. The detailed results for the initial bounds in Tables 6, 10, 11, and 12 show that it is reasonable to calculate all subsystems, as even large subsystems improve the (aggregated) bounds on the buffer capacities.

The column "Max. deviation from goal throughput" depicts the results of a throughput evaluation for the different optimal buffer allocations obtained from the different samples. The throughput evaluation is conducted with 10 new samples of 1,000,000 workpieces for each category of instances. The column shows the largest relative downward deviation of all optimal allocations if the goal throughput was not reached and the smallest relative upward deviation if it was reached. The deviation for each buffer allocation is shown in Tables 6, 10, 11, and 12. Very small downward and upward deviations are denoted as −0.00 and 0.00, respectively.

The maximum downward deviation obtained from all 80 optimization runs is only 0.36 %. Altogether, this shows that the Benders Decomposition approach with combinatorial cuts and initial bounds is able to optimize flow lines with generally distributed processing times quite well.

4.5 Correlated processing times

This experiment investigates the impact of statistical dependency on the optimal buffer allocation. Inman (1999) points out that statistical dependency of processing times, i.e., workpiece-dependent processing times at each station, occurs for example in the automotive industry when two- and four-door models are manufactured on the same line. We model this by generating processing times from Erlang-4 distribution with different rates for the two different types of workpieces. The rate corresponding to a workpiece of type 1 is set to 0.5, while the rate for workpieces of type 2 is 0.25. We assume that the probability that a workpieces is of type 2 is 20%. Non-listed parameters remain as in the base case (Table 4). We compare the results to allocations obtained from instances generated by Generalized Erlang distribution based on identical parameters. This corresponds to the case where correlation is neglected and approximated by independent identically distributed processing times. Generalized Erlang distribution may be interpreted as a random decision on the type for each processed workpiece at each station.

All instances were solved in less than 15 min. Further computational results are given in Table 7. Columns 2–4 correspond to the instances with correlation in processing times and the last three columns to the instances with Generalized Erlang distribution. The results show that the instances with Generalized Erlang distribution underestimate the throughput and therefore allocate more buffer spaces than necessary, mainly around the bottleneck. For the instances under investigation, on average 26 % additional buffer spaces were allocated. In conclusion, the approximation of correlated processing times by identical independently distributed processing times leads to sub-

Table 7 Detailed results (correlated processing times)

Sample	Correlation			Generalized Erlang		
	Total buffer capacities	Optimal allocation	Max. dev. from TH* (%)	Total buffer capacities	Optimal allocation	Max. dev. from TH* (%)
1	30	5, 4, 7, 7, 4, 3	−0.04	38	4, 5, 11, 9, 5, 4	0.07
2	30	3, 6, 6, 7, 4, 4	0.06	37	5, 6, 8, 7, 6, 5	−0.15
3	30	3, 5, 7, 6, 5, 4	0.09	38	4, 7, 7, 10, 5, 5	−0.00
4	29	4, 4, 7, 5, 6, 3	−0.29	37	4, 6, 9, 8, 6, 4	−0.06
5	29	4, 4, 6, 7, 5,3	−0.14	37	4, 6, 9, 8, 6, 4	−0.06
6	30	3, 6, 6, 6, 5, 4	0.03	37	4, 7, 8, 8, 5, 5	−0.12
7	30	3, 5, 7, 6, 5, 4	0.09	37	4, 7, 8, 8, 5, 5	−0.12
8	30	4, 5,6, 7, 5, 3	0.08	38	3, 7, 9, 9, 5, 5	−0.04
9	29	4, 4,7, 6, 4, 4	−0.13	37	4, 6, 8, 9, 6, 4	−0.06
10	30	3, 5, 7, 8, 3, 4	−0.02	38	4, 7, 8, 8, 6, 5	0.03

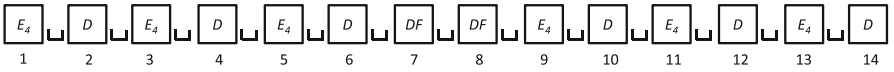


Fig. 11 Setting of the 14-station line

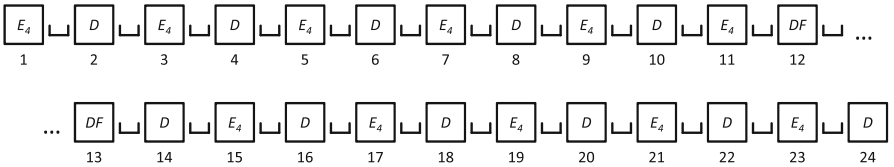


Fig. 12 Setting of the 24-station line

stantial misallocation of buffer spaces. Therefore, it is important that correlations are considered in the solution approach, as it is possible with our approach.

4.6 Long lines with reliable and unreliable stations

This experiment is devoted to long lines comprising 14 and 24 stations, respectively, some of which are reliable and others are unreliable (see Figs. 11, 12).

Reliable stations have Erlang-4-distributed (E_4) or deterministic (D) processing times, both with rate 0.5. Unreliable stations (DF) have deterministic processing times with rate 0.5 and exponentially distributed times to failure (TTF) and times to repair (TTR). The mean TTF and the mean TTR are chosen such that stations in the middle of the line, i.e., 7 and 8 in the line with 14 stations and 12 and 13 in the line with 24 stations, form the bottlenecks of the line. Non-listed parameters remain as in the base case (Table 4).

Table 8 Detailed results ($S = 14$)

Sample	Total buffer capacities	Optimal allocation	Computation times (s)		Max. dev. from TH* (%)
			Subsystems	Total	
TTF = 10; TTR = 4					
1	13	0, 0, 0, 1, 2, 0, 7, 2, 1, 0, 0, 0, 0	178	234	−0.74
2	13	0, 0, 0, 1, 0, 2, 7, 2, 1, 0, 0, 0, 0	138	140	−0.90
3	14	0, 0, 0, 0, 0, 4, 7, 2, 1, 0, 0, 0, 0	384	393	−0.67
4	14	1, 0, 0, 0, 3, 0, 7, 2, 1, 0, 0, 0, 0	295	298	−0.61
5	14	1, 0, 0, 1, 2, 0, 6, 2, 0, 1, 0, 1, 0	276	279	−0.61
6	14	0, 0, 2, 0, 1, 1, 7, 2, 1, 0, 0, 0, 0	320	325	−0.54
7	14	0, 0, 0, 2, 2, 0, 6, 3, 1, 0, 0, 0, 0	387	466	−0.42
8	14	0, 0, 1, 0, 3, 0, 7, 2, 1, 0, 0, 0, 0	371	375	−0.40
9	14	0, 0,1, 1, 1, 1, 6, 2, 2, 0, 0, 0, 0	324	326	−0.49
10	13	0, 0, 0, 1, 2, 0, 7, 2, 1, 0, 0, 0, 0	223	257	−0.74
∅			290	309	−0.61
TTF = 5; TTR = 2					
1	8	0, 0, 1,0, 1, 0, 3, 2, 0, 1, 0, 0, 0	47	49	−0.26
2	8	0, 0, 1, 0, 1, 0, 3, 2, 0, 1, 0, 0, 0	42	44	−0.26
3	8	0, 0, 0, 0, 1, 1, 4, 1, 0, 1, 0, 0, 0	52	54	0.04
4	8	0, 0, 1, 0, 1, 0, 3, 2, 0, 1, 0, 0, 0	42	44	−0.26
5	8	0, 0, 1, 0, 1, 0, 3, 2, 0, 1, 0, 0, 0	52	54	−0.26
6	8	0, 0, 1, 0, 1, 0, 3, 2, 0, 1, 0, 0, 0	52	54	−0.26
7	8	0, 0, 1, 0, 1, 0, 3, 2, 0, 1, 0, 0, 0	49	50	−0.26
8	8	0, 0, 1, 0, 1, 0, 3, 2, 0, 1, 0, 0, 0	52	54	−0.26
9	8	0, 0, 1, 0, 1, 0, 3, 2, 0, 1, 0, 0, 0	46	48	−0.26
10	8	0, 0, 0, 0, 0, 2, 4, 2, 0, 0, 0, 0, 0	48	49	0.15
∅			48	50	−0.01

Tables 8 and 9 contain the results of the third experiment on long lines. The algorithm solves instances with 14 stations within 310 s on average for TTF = 10 and TTR = 4. If TTF = 5 and TTR = 2, the algorithm takes 50 s on average. For the line with 24 stations, 14 h on average are required to prove the optimal solution for TTF = 10 and TTR = 4. The instances with TTF = 5 and TTR = 2 can be solved within 10 min on average. The algorithm spends most of the time to calculate the results for the subsystems (93–98 % of the total time). Altogether, under consideration of the strategic nature of the Buffer Allocation Problem, the algorithm is able to optimize long lines in acceptable time.

The majority of the buffer spaces (in many cases half of the total allocated capacities) is allocated between the bottleneck stations. At the beginning and at the end of the line, zero or only few buffer spaces are required. The last column

Table 9 Detailed results ($S = 24$)

Sample	Total buffer capacities	Optimal allocation	Computation times (s)		Max. dev. from TH* (%)
			Subsystems	Total	
TTF = 10; TTR = 4					
1	17	0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 3, 8, 0, 3, 0, 0, 0, 1, 0, 0, 0, 0, 0	53,803	54,189	−0.68
2	17	0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 2, 8, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0	48,436	50,519	−0.56
3	17	0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 3, 8, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,0	24,432	25,999	−0.65
4	17	0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 2, 6, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0	49,414	49,818	−0.68
5	17	0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 3, 8, 1, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0	47,830	48,913	−0.69
6	17	0, 0, 0, 0, 1, 0, 0, 0, 0, 2, 3, 7, 0, 2, 1, 0, 1, 0, 0, 0, 0, 0, 0	61,035	61,064	−0.54
7	17	0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 2, 8, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0	69,225	69,234	−0.43
8	17	0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 2, 6, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0	52,257	52,648	−0.68
9	17	0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 3, 7, 0, 3, 2, 0, 0, 0, 0, 0, 0, 0	47,098	48,703	−0.66
10	17	0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 3, 7, 1, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0	52,572	53,513	−0.71
∅			50,610	51,460	−0.63
TTF = 5; TTR = 2					
1	9	0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 3, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0	601	616	−0.65
2	9	0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 4, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0	737	787	−0.36
3	9	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 4, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	530	538	−0.48
4	9	0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 4, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0	513	528	−0.36
5	9	0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 4, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0	574	613	−0.36
6	9	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 4, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	751	767	−0.48
7	9	0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 4, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0	581	596	−0.33
8	9	0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 4, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0	697	713	−0.32
9	9	0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 4, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0	518	521	−0.36
10	9	0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 4, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0	505	511	−0.18
∅			601	619	−0.38

in each table depicts the results of a throughput evaluation for the different optimal buffer allocations obtained from the different samples. The throughput evaluation is conducted with 10 new samples of 1,000,000 workpieces for each category of instances. The column shows the largest relative downward deviation of all optimal allocations if the goal throughput was not reached and the smallest relative upward deviation if it was reached. The maximum deviation obtained from all optimization runs is only 0.61 % for the 14-station line and 0.62 % for the 24-station line.

5 Conclusion and further research

In this paper, we develop a Benders Decomposition approach that is able to optimally solve the BAP with respect to an underlying sample. This approach divides the original problem into a master problem and a subproblem, which are both solved iteratively by exchanging information via cuts. We compared two types of cuts, classical feasibility cuts and combinatorial feasibility cuts. Our numerical study shows that the application of combinatorial cuts leads to substantial reductions in the computation time. Furthermore, we develop initial lower bounds based on the iterative solutions of subsystems for the original line. This approach is able to optimally allocate buffer spaces in long lines with arbitrary distributions of processing times, times to failure, and repair times within a reasonable amount of time. The numerical study also reveals that correlation effects in processing times have a significant effect, as the optimal buffer allocation is highly influenced. This demonstrates the necessity for flexible solution approaches, as the sample-based mathematical programming formulations.

Further research should be directed towards improving the computation times for lines with more stations. This may be performed by the analysis of additional bounds or by the development of a problem-specific branch and bound method. Additionally, the approach could be extended to more complex systems, such as flow lines with closed loops or several product types.

6 Appendix: Detailed results for Erlang-k and Cox-2 distributed instances

See Tables 10, 11, and 12.

Table 10 Detailed results (Cox-2 distribution, $S = 5$)

Sample	SCV	Optimal allocation	Max. dev. from TH* (%)	Initial bounds								
				$i = 2$				$i = 3$			$i = 4$	
				b_1	b_2	b_3	b_4	$\sum_{j=1}^{2} b_j$	$\sum_{j=2}^{3} b_j$	$\sum_{j=3}^{4} b_j$	$\sum_{j=1}^{3} b_j$	$\sum_{j=2}^{4} b_j$
1, 2	1.0	7, 8, 8, 6	−0.22	3	5	5	3	12	13	12	21	21
3, 7	1.0	6, 8, 9, 6	−0.12	3	5	5	3	12	13	12	21	21
4	1.0	6, 9, 8, 6	−0.12	3	5	5	3	12	13	12	21	21
5	1.0	7, 7, 10, 6	0.04	3	5	5	3	12	13	12	21	21
6, 9, 10	1.0	5, 10, 10, 5	0.12	3	5	5	3	12	13	12	21	21
8	1.0	5, 10, 8, 6	−0.21	3	5	4	3	12	12	11	21	20
1	2.0	12, 18, 18, 13	0.01	6	9	9	5	24	26	24	44	43
2	2.0	11, 18, 18, 13	−0.26	5	9	9	5	24	26	24	43	43
3	2.0	13, 20, 17, 12	0.12	6	9	9	5	25	27	25	44	44
4	2.0	11, 18, 20, 12	−0.09	5	9	9	5	24	26	24	43	43
5	2.0	13, 19, 15, 14	−0.17	5	9	9	5	24	26	24	43	43
6	2.0	13, 17, 18, 13	−0.02	5	9	9	6	24	26	24	43	44
7	2.0	13, 18, 18, 12	0.00	6	9	9	5	25	27	24	44	44
8	2.0	14, 17, 18, 12	−0.05	5	9	9	6	24	26	25	43	44
9	2.0	13, 16, 20, 13	0.09	5	9	9	5	24	26	25	44	44
10	2.0	12, 19, 17, 14	0.16	5	9	9	5	24	26	24	44	43

Table 11 Detailed results (Erlang-k distribution, $S = 7$)

Sample	SCV	Optimal allocation	Max. dev. from TH* (%)	Initial bounds										
				$i = 2$						$i = 3$				
				b_1	b_2	b_3	b_4	b_5	b_6	$\sum_{j=1}^{2} b_j$	$\sum_{j=2}^{3} b_j$	$\sum_{j=3}^{4} b_j$	$\sum_{j=4}^{5} b_j$	$\sum_{j=5}^{6} b_j$
1, 4, 10	0.25	1, 2, 2, 3, 1, 1	0.21	**1**	1	1	1	**1**	**1**	2	3	3	3	**2**
2, 5, 7	0.25	1, 2, 2, 2, 1, 2	0.14	**1**	1	1	1	**1**	1	2	3	3	**3**	2
3, 6	0.25	1, 1, 3, 3, 1, 1	−0.05	**1**	**1**	1	1	**1**	**1**	**2**	3	3	3	**2**
8, 9	0.25	1, 1, 3, 2, 2, 1	0.26	**1**	**1**	1	1	1	**1**	**2**	3	3	3	2
1, 3, 4, 7–10	0.5	2, 5, 4, 4, 4, 3	−0.00	1	1	2	2	1	1	4	6	6	6	4
2	0.5	2, 5, 5, 4, 3, 3	−0.04	1	1	2	2	1	1	4	6	6	5	4
5	0.5	4, 3, 4, 5, 4, 2	−0.09	1	1	2	2	1	1	4	5	6	5	4
6	0.5	2, 4, 5, 5, 4, 2	0.03	1	1	2	2	1	1	4	6	6	6	4

Table 11 continued

Sample	SCV	Optimal allocation	Max. dev. from TH* (%)	Initial bounds								
				$i = 4$				$i = 5$			$i = 6$	
				$\sum_{j=1}^{3} b_j$	$\sum_{j=2}^{4} b_j$	$\sum_{j=3}^{5} b_j$	$\sum_{j=4}^{6} b_j$	$\sum_{j=1}^{4} b_j$	$\sum_{j=2}^{5} b_j$	$\sum_{j=3}^{6} b_j$	$\sum_{j=1}^{5} b_j$	$\sum_{j=2}^{6} b_j$
1, 4, 10	0.25	1, 2, 2, 3, 1, 1	0.21	4	5	5	4	6	6	6	8	8
2, 5, 7	0.25	1, 2, 2, 2, 1, 2	0.14	4	5	**5**	4	6	6	7	**8**	8
3, 6	0.25	1, 1, 3, 3, 1, 1	−0.05	4	5	5	4	6	6	6	8	8
8, 9	0.25	1, 1, 3, 2, 2, 1	0.26	4	5	5	4	**7**	6	7	8	8
1, 3, 4, 7–10	0.5	2, 5, 4, 4, 4, 3	−0.00	9	10	10	9	14	14	14	18	18
2	0.5	2, 5, 5, 4, 3, 3	−0.04	9	10	10	9	14	14	14	18	18
5	0.5	4, 3, 4, 5, 4, 2	−0.09	9	10	10	9	14	14	14	18	18
6	0.5	2, 4, 5, 5, 4, 2	0.03	9	10	10	9	14	14	14	18	18

Table 12 Detailed results (Cox-2 distribution, $S = 7$)

Sample	SCV	Optimal allocation	Max. dev. from TH* (%)	Initial bounds										
				$i = 2$						$i = 3$				
				b_1	b_2	b_3	b_4	b_5	b_6	$\sum_{j=1}^{2} b_j$	$\sum_{j=2}^{3} b_j$	$\sum_{j=3}^{4} b_j$	$\sum_{j=4}^{5} b_j$	$\sum_{j=5}^{6} b_j$
1	1.0	6, 7, 10, 12, 7, 5	0.11	3	3	5	5	3	3	9	12	13	12	9
2	1.0	7, 7, 8, 10, 9, 6	0.05	3	3	5	5	3	3	9	12	13	12	9
3	1.0	6, 7, 10, 10, 7, 6	0.03	3	3	5	5	3	3	9	12	13	12	9
4	1.0	6, 7, 10, 10, 7, 6	0.03	3	3	5	5	3	3	9	12	13	12	9
5	1.0	6, 7, 11, 9, 7, 6	−0.04	3	3	5	5	3	3	9	12	13	12	9
6	1.0	7, 8, 8, 9, 8, 7	0.06	3	3	5	5	3	3	9	12	13	12	9
7	1.0	6, 7, 11, 9, 7, 6	−0.04	3	3	5	5	3	3	9	12	13	12	9
8	1.0	6, 8, 9, 10, 7, 6	0.01	3	3	5	5	3	3	9	12	13	12	9
9	1.0	5, 8, 10, 10, 6, 8	0.04	3	3	5	5	3	3	9	12	13	12	9
10	1.0	6, 7, 10, 9, 7, 7	−0.04	3	3	5	5	3	3	9	12	13	12	9
1	2.0	13, 18, 18, 19, 16, 13	−0.13	5	5	9	9	5	5	18	24	26	24	18
2	2.0	13, 16, 19, 20, 15, 14	−0.09	6	5	9	9	5	6	18	24	26	24	18
3	2.0	13, 16, 19, 20, 17, 13	0.07	6	6	9	9	6	6	18	25	26	25	18
4	2.0	12, 18, 20, 19, 14, 14	−0.15	5	6	9	9	5	5	18	24	26	24	18
5	2.0	13, 15, 20, 19, 16, 13	−0.20	6	5	9	9	5	5	18	24	25	24	18
6	2.0	12, 16, 19, 20, 18, 13	0.03	5	6	9	9	5	5	18	25	27	25	18
7	2.0	13, 15, 19, 19, 16, 13	−0.36	5	5	9	9	5	5	18	23	26	24	18
8	2.0	11, 17, 20, 20, 16, 13	−0.09	5	6	9	9	5	6	18	24	26	24	19
9	2.0	12, 16, 20, 20, 16, 13	−0.05	5	5	9	9	5	5	18	24	26	24	18
10	2.0	12, 17, 20, 19, 16, 13	−0.07	6	5	9	9	6	6	18	24	26	24	18

Table 12 continued

Sample	SCV	Optimal allocation	Max. dev. from TH* (%)	Initial bounds								
				$i = 4$				$i = 5$			$i = 6$	
				$\sum_{j=1}^{3} b_j$	$\sum_{j=2}^{4} b_j$	$\sum_{j=3}^{5} b_j$	$\sum_{j=4}^{6} b_j$	$\sum_{j=1}^{4} b_j$	$\sum_{j=2}^{5} b_j$	$\sum_{j=3}^{6} b_j$	$\sum_{j=1}^{5} b_j$	$\sum_{j=2}^{6} b_j$
1	1.0	6, 7, 10, 12, 7, 5	0.11	19	21	21	20	29	30	30	38	38
2	1.0	7, 7, 8, 10, 9, 6	0.05	20	21	21	20	29	29	29	38	38
3	1.0	6, 7, 10, 10, 7, 6	0.03	20	21	21	19	29	30	29	38	38
4	1.0	6, 7, 10, 10, 7, 6	0.03	20	21	21	19	29	29	29	38	38
5	1.0	6, 7, 11, 9, 7, 6	−0.04	19	21	21	20	29	29	29	38	38
6	1.0	7, 8, 8, 9, 8, 7	0.06	19	21	21	20	29	30	29	38	38
7	1.0	6, 7, 11, 9, 7, 6	−0.04	20	21	21	19	29	29	29	38	38
8	1.0	6, 8, 9, 10, 7, 6	0.01	19	21	21	20	29	29	29	38	38
9	1.0	5, 8, 10, 10, 6, 8	0.04	19	21	21	20	29	30	29	38	38
10	1.0	6, 7, 10, 9, 7, 7	−0.04	19	21	21	19	29	29	29	38	38
1	2.0	13, 18, 18, 19, 16, 13	−0.13	40	43	43	41	60	61	61	79	78
2	2.0	13, 16, 19, 20, 15, 14	−0.09	40	43	43	41	60	61	61	78	79
3	2.0	13, 16, 19, 20, 17, 13	0.07	41	43	44	41	61	62	62	80	80
4	2.0	12, 18, 20, 19, 14, 14	−0.15	41	44	43	40	61	61	61	79	79
5	2.0	13, 15, 20, 19, 16, 13	−0.20	40	42	43	40	60	61	60	79	78
6	2.0	12, 16, 19, 20, 18, 13	0.03	41	44	44	41	61	62	62	80	80
7	2.0	13, 15, 19, 19, 16, 13	−0.36	39	43	43	40	60	60	60	78	78
8	2.0	11, 17, 20, 20, 16, 13	−0.09	40	43	43	41	60	61	61	79	80
9	2.0	12, 16, 20, 20, 16, 13	−0.05	40	43	43	41	60	61	61	79	79
10	2.0	12, 17, 20, 19, 16, 13	−0.07	40	43	43	41	61	61	61	79	79

References

Alfieri A, Matta A (2012) Mathematical programming formulations for approximate simulation of multistage production systems. Eur J Oper Res 219(3):773–783

Alfieri A, Matta A (2013) Mathematical programming time-based decomposition algorithm for discrete event simulation. Eur J Oper Res 231(3):557–566

Bai L, Rubin PA (2009) Combinatorial benders cuts for the minimum tollbooth problem. Oper Res 57(6):1510–1522

Benders J (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4(1):238–252

Burman M, Gershwin SB, Suyematsu C (1998) Hewlett-packard uses operations research to improve the design of a printer production line. Interfaces 28(1):24–36

Buzacott JA, Shanthikumar JG (1993) Stochastic models of manufacturing systems, vol 4. Prentice Hall, Englewood Cliffs

Caramanis M (1987) Production system design: A discrete event dynamic system and generalized benders' decomposition approach. Int J Prod Res 25(8):1223–1234

Chan WKV, Schruben L (2008) Optimization models of discrete-event system dynamics. Oper Res 56(5):1218–1237

Codato G, Fischetti M (2006) Combinatorial benders cuts for mixed-integer linear programming. Oper Res 54(4):756–766

Colledani M, Ekvall M, Lundholm T, Moriggi P, Polato A, Tolio T (2010) Analytical methods to support continuous improvements at scania. Int J Prod Res 48(7):1913–1945

Cooke RM, Bosma A, Härte F (2005) A practical model of heineken's bottle filling line with dependent failures. Eur J Oper Res 164(2):491–504

Dallery Y, Gershwin SB (1992) Manufacturing flow line systems: a review of models and analytical results. Queueing Syst 12(1):3–94

Demir L, Tunali S, Eliiyi DT (2014) The state of the art on buffer allocation problem: a comprehensive survey. J Intell Manuf 25(3):371–392

Diamantidis A, Papadopoulos C (2004) A dynamic programming algorithm for the buffer allocation problem in homogeneous asymptotically reliable serial production lines. Math Probl Eng 2004(3):209–223

Gershwin SB, Schor JE (2000) Efficient algorithms for buffer space allocation. Ann Oper Res 93(1):117–144

Gürkan G (2000) Simulation optimization of buffer allocations in production lines with unreliable machines. Ann Oper Res 93(1–4):177–216

Helber S, Schimmelpfeng K, Stolletz R, Lagershausen S (2011) Using linear programming to analyze and optimize stochastic flow lines. Ann Oper Res 182(1):193–211

Hillier FS, So KC, Boling RW (1993) Toward characterizing the optimal allocation of storage space in production line systems with variable processing times. Manag Sci 39(1):126–133

Hillier MS (2000) Characterizing the optimal allocation of storage space in production line systems with variable processing times. IIE Transactions 32(1):1–8

Inman RR (1999) Empirical evaluation of exponential and independence assumptions in queueing models of manufacturing systems. Prod Oper Manag 8(4):409–432

Levantesi R, Matta A, Tolio T (2001) A new algorithm for buffer allocation in production lines. In: Proceedings of the 3rd Aegean international conference on design and analysis of manufacturing systems, pp 19–22

Li J (2013) Continuous improvement at toyota manufacturing plant: applications of production systems engineering methods. Int J Prod Res 51(23–24):7235–7249

Li J, Meerkov SM (2009) Production Systems Engineering. Springer Science+ Business Media LLC, Boston

Liberopoulos G, Tsarouhas P (2005) Reliability analysis of an automated pizza production line. J Food Eng 69(1):79–96

Lutz CM, Davis KR, Sun M (1998) Determining buffer location and size in production lines using tabu search. Eur J Oper Res 106(2):301–316

MacGregor Smith J, Cruz F (2005) The buffer allocation problem for general finite buffer queueing networks. IIE Trans 37(4):343–365

Matta A (2008) Simulation optimization with mathematical programming representation of discrete event systems. In: Proceedings of the 2008 winter simulation conference, Miami, pp 1393–1400

Matta A, Chefson R (2005) Formal properties of closed flow lines with limited buffer capacities and random processing times. In: Proceedings of the European simulation and modelling conference, Portugal, pp 190–194

Powell SG, Pyke DF (1996) Allocation of buffers to serial production lines with bottlenecks. IIE Trans 28(1):18–29

Saliby E (1990a) Descriptive sampling: a better approach to monte carlo simulation. J Oper Res Soc 41(12):1133–1142

Saliby E (1990b) Understanding the variability of simulation results: an empirical study. J Oper Res Soc 41(4):319–327

Schruben LW (2000) Mathematical programming models of discrete event system dynamics. In: Proceedings of the 32nd conference on winter simulation, Orlando, pp 381–385

Spinellis DD, Papadopoulos CT (2000) A simulated annealing approach for buffer allocation in reliable production lines. Ann Oper Res 93(1–4):373–384

Stolletz R, Weiss S (2013) Buffer allocation using exact linear programming formulations and sampling approaches. In: 7th IFAC conference on manufacturing modelling, management, and control, St. Petersburg, pp 1435–1440

Yamashita H, Altiok T (1998) Buffer capacity allocation for a desired throughput in production lines. IIE Trans 30(10):883–892