UNIVERSITY OF MANNHEIM

# Optimization of buffer allocations in stochastic flow lines

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Wirtschaftswissenschaften
der Universität Mannheim

vorgelegt von

Sophie Weiss
Mannheim

Dekan: *Dr. Jürgen M. Schneider*

Referent: *Prof. Dr. Raik Stolletz*

Korreferent: *Prof. Dr. Moritz Fleischmann*

Tag der mündlichen Prüfung: *02. Oktober 2015*

*To my grandparents Lotti and Hans*

# Summary

The design of flow lines is an important task in practice which is currently not supported sufficiently by the literature. An important question is how to allocate buffer capacities within the flow line. This thesis develops exact solution methods which efficiently optimize the buffer allocation in flow lines under general assumptions. The first essay provides an overview on existing literature in the field of buffer optimization. A classification scheme is developed to facilitate the comparison of different algorithms. The second essay investigates exact mixed-integer programming approaches to calculate optimal buffer capacities. These approaches allow the consideration of many real-world features that have not yet been covered in the literature. However, they cannot be applied in practice because of their low computational performance. In the third essay, an exact algorithm which uses Benders Decomposition is proposed in order to overcome the shortcomings of the mixed-integer programs. Lower bounds on the buffer capacities are developed to reduce the solution space. The key result is that this algorithm leads to good computational performance when optimizing lines under general assumptions. The fourth essay investigates the impact of limited supply on the optimal buffer allocation. An exact solution method based on individual lower bounds and rule-based generation of candidate allocations is developed. The major finding is that a limited supply not only leads to an increase in the optimal buffer capacities, but may also decrease the throughput of the line to such an extent that the throughput goals of the company cannot be met, even by adding additional buffer capacities.
Further research should concentrate on extending the proposed solution approaches to take into account more complex systems, such as flow lines with closed loops or several product types. In addition, it is desirable to develop a model which allows for simultaneous optimization of the buffer capacities and the raw material supply.

# Contents

# List of Figures

# List of Tables

# 1. Introduction

Flow lines are manufacturing systems which enable high production volumes for relatively low costs. They are typically to be found in the automotive and in the food industry, among others. A flow line consists of a number of machines in series with a fixed sequence of the processing steps. Such lines are of their nature subject to unpredictable, i.e., stochastic, interruptions. On the one hand, these may be caused by unreliable machinery suffering from inevitable breakdowns and subsequent repairs of random duration. On the other hand, variability in processing time may originate from human operators (Tempelmeier, 2003). Both influences lead to interruptions of the material flow and therefore reduce the line's performance. In-process storages, so-called buffers, can mitigate the impact of these stochastic influences by decoupling the machines to a certain extent. Hence, the material flow does not necessarily have to be synchronized, i.e., the workpieces move independently. Consequently, buffers can also facilitate the production of variants or similar products as long as the processing steps and the processing times are comparable (Buzacott and Shanthikumar, 1993, p.2). Thus buffer capacities do not only increase machine utilization and therefore the throughput, but also the flexibility of the flow lines. However, the installation and provision of such capacities is related to investments and costs for additional storage (Gershwin and Schor, 2000).

Both the output and the costs of such lines may have a significant influence on the competitiveness of the plant or even the company as a whole (Burman et al., 1998; Alden et al., 2006). In particular, low or misallocated buffer capacities may cause low output due to inefficient machine utilization, whereas high buffer capacities lead to high investments and high costs originating from excessive in-process inventory and the amount of floor space required (Gershwin and Schor, 2000). The Buffer Allocation Problem attempts to balance out these counteracting targets.

When optimizing the buffer allocation in practice, two scenarios are distinguished. On the one hand, if demand exceeds supply, increasing the throughput of the line generates additional output which can be converted into additional sales. In practice, overtime is often used in the long run to increase production capacity should the production targets not be met (Patchong et al., 2003; Alden et al., 2006). However in

this case, the overtime and its related costs can be avoided by allocating additional buffer capacities to increase the productivity. On the other hand, if supply exceeds demand, decreasing costs while preserving the profit will increase profitability. Reallocating and reducing buffer capacities adequately decrease those costs related to storage and maintain the actual performance of the line. The following examples underline the tremendous impact of effectively designed production lines. Burman et al. (1998) point out that the optimization of the printer production at Hewlett-Packard raised the revenues by $280 million. Alden et al. (2006) report savings and additional revenues of $2.1 billion within 20 years at General Motors obtained from buffer optimization and other performance improvements. PSA Peugeot Citroën reduced overtime and re-designed their production lines resulting in $130 million of additional profit, which corresponds to about 6.5% of the total profit in 2001 (Patchong et al., 2003). In the food industry, stock deterioration and costs incurred from scrapping must also be considered. Liberopoulos and Tsarouhas (2002) describe the case of a Greek croissant manufacturer. Optimized buffer capacities led to an additional profit of $19,150 per week, made up of additional revenue from sales as a result of increasing output and lower costs from reduced scrapping and reduced overtime.

The Buffer Allocation Problem is difficult to solve for several reasons. To begin with, the causes for low throughput are often not obvious to plant workers and managers (Alden et al., 2006). Moreover, the line is in general highly sensitive to changes in its characteristics and the related data (Tempelmeier, 2003). Therefore, the problem cannot be solved by intuition or experience. Finally, combinatorial complexity emerges from the number of candidate allocations.

To conclude, the optimization of buffer allocations in stochastic flow lines is a well-established problem which has been investigated for several decades. Yet existing exact approaches are based on restrictive assumptions. They are therefore impracticable for solving real-world problems. Optimizing the buffer allocation efficiently under general assumptions requires approximative solution approaches which can lead to unsatisfactory inaccuracies.

This illustrates the need for exact and efficient solution methods that are applicable under general assumptions.

This dissertation presents an overview of the literature on the Buffer Allocation Problem and investigates how the consideration of realistic features impacts on the optimal buffer allocation. Furthermore, it introduces new algorithms which are capable of solving the Buffer Allocation Problem in such a way as to overcome the drawbacks of the existing approaches.

Chapter 2 introduces a classification scheme of the characteristics of a flow line and the decision problems which arise in the context of the Buffer Allocation Problem. This work has been motivated by the observation that the underlying assumptions and solution characteristics are often reported on insufficiently or not at all. The literature covering unreliable machines is reviewed and categorized in accordance with the developed classification scheme. Here we identify common assumptions, existing test instances, and solution approaches. This facilitates the comparison of existing and new algorithms for the Buffer Allocation Problem. This article was written jointly with Justus Arne Schwarz and Raik Stolletz[1].

The study in Chapter 3 elaborates on an exact solution method for the Buffer Allocation Problem, which allows the incorporation of realistic features of the line. When optimizing the buffer allocation, realistic line characteristics, such as generally distributed processing times, times to failure, or times to repair are often neglected. If these features are considered in the literature, the Buffer Allocation Problem is merely solved heuristically. In this chapter, mixed-integer programming in combination with sampling is applied. The sampling approaches thereby cover the stochastic effects of the system by replacing the random variables by sampled realizations. Such procedures correspond to a simulation of the flow of a number of workpieces through the system. There are different sampling methods available to generate the realizations of the random variables. An important issue is how to select the required number of workpieces, i.e., the sample size, which will properly model the line and lead to robust results, while preserving acceptable computation times. We conduct an extensive numerical study to investigate the accuracy and computational performance of different sample sizes and sampling methods. To this end, we identify appropriate sample sizes for the test cases under investigation. Moreover, several mixed-integer programming formulations and standard solvers are compared in order to identify the most efficient combination with respect to the computation time. This is a first step towards the development of a flexible and efficient solution algorithm for the Buffer Allocation Problem. This article was written jointly with Raik Stolletz[2].

The fourth chapter presents an efficient and exact approach for the determination of optimal buffer allocations. The in-depth analysis of the performance of the mixed-integer programming formulations presented in Chapter 2 reveals that these for-

[1] Weiss, S., J. A. Schwarz, and R. Stolletz (2015). Buffer Allocation Problems for stochastic flow lines with unreliable machines. In Proceedings of the 10th conference on stochastic models of manufacturing and service operations, Volos, Greece, pp. 271-277

[2] Stolletz, R. and S. Weiss (2013). Buffer allocation using exact linear programming formulations and sampling approaches. In Manufacturing Modelling, Management, and Control, Volume 7(1), St. Petersburg, Russia, pp. 1435–1440.

mulations require very long computation times and are often not solvable because of the high memory consumption. To reduce the solution space of the problem, we introduce bounds on the buffer capacities. To this end, we iteratively optimize subsystems, each of which consists of a subset of the line. We prove that these subsystems produce a higher output for a given buffer capacity than the original line. As a consequence, optimizing the buffer allocation in such subsystems provides lower bounds on the total capacity of the respective buffers in the original line. This concept is independent of the applied solution approach and can therefore be used in any algorithm to solve the Buffer Allocation Problem. Furthermore, we apply a Benders Decomposition, which divides the sample-based mixed-integer program into two parts. First, candidate allocations are generated by an integer program. Secondly, they are evaluated using a sample-based simulation algorithm. Both parts are connected by adding cuts to the integer program, which are generated from information obtained by the evaluation routine. We show that the application of the lower bounds in combination with the Benders Decomposition approach significantly reduces the computation times. Additionally, we investigate the impact of correlations in processing times on the optimal buffer allocation. Our study reveals that ignoring correlations leads to an overestimation of the required buffer capacities and hence to additional costs. This article was written jointly with Raik Stolletz[3].

Chapter 5 investigates the impact of a limited supply on the optimal buffer capacities. Our research is motivated by the observation that the interplay of supply and demand of the line has up to now been neglected, i.e., system state-independent arrivals are assumed. In general, order policies manage the supply of workpieces to the line based on the material consumption. That is, orders are only placed if the system signals that an insufficient amount of material is available. We therefore use order policies to model the supply of the flow line. To efficiently solve the resulting Buffer Allocation Problem, we develop a two-step rule-based local search algorithm. The first step generates candidate allocations by applying certain rules. The second step evaluates these candidates and returns this information to the first step. In contrast to the Benders Decomposition algorithm, this algorithm uses information from the evaluations to guide the search for the optimal allocation. Moreover, we propose individual lower bounds for the buffer capacities which are applied within the rule-based local search algorithm. A mathematical programming model is developed to calculate these. We demonstrate that the application of the

[3]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.

individual lower bounds in combination with the rule-based local search algorithm significantly reduces the computation times. In addition, we show that the choice of the order policy and its parameters greatly influences the optimal total buffer capacity and its allocation. The lack of material induced by the limited supply is compensated by additional buffer capacities. These capacities allow workpieces to already enter the line which subsequently triggers earlier replenishment or higher order quantities. However, after a certain point, additional buffer capacities cannot cope with the interplay between stochastic effects and limited supply. This leads to a decrease in the throughput of the line, which cannot be compensated for. This article was written jointly with Andrea Matta and Raik Stolletz[4].

[4] Weiss, S., A. Matta, and R. Stolletz (2015). Optimization of buffer allocations in flow lines with limited supply. Working paper.

# 2. Buffer Allocation Problems for stochastic flow lines with unreliable machines

*Co-authors:*

**Justus Arne Schwarz**
Chair of Production Management, Business School, University of Mannheim, Germany

**Raik Stolletz**
Chair of Production Management, Business School, University of Mannheim, Germany

*Published in:*

Proceedings of the 10th conference on stochastic models of manufacturing and service operations, Volos, Greece, 2015, pages 271-277

*Abstract:*

The Buffer Allocation Problem in serial production lines is solved for different objectives, constraints, and assumptions. The aim of this work is to characterize analyzed production lines with unreliable machines and the underlying decision problems. We investigate unreliable serial lines with finite intermediate buffers and a single machine per station that processes discrete material. Moreover, we review existing solution approaches.

## 2.1. Introduction

Flow lines process workpieces sequentially on multiple stations. These production systems usually have a finite buffer capacity and are frequently used in manufacturing, in particular in the automotive industry (Tempelmeier, 2003; Li, 2013). They often experience random processing times, stochastic failures, and successive repairs. This leads to blocking and starvation which reduce the throughput of the line. A station starves if it cannot produce due to a lack of material in the upstream buffer whereas a blocked machine stops production due to a full downstream buffer. The choice of the total buffer capacity and its allocation between machines is a key design decision. This is because buffer capacities are associated with the costs of the buffer itself and the related work-in-process inventory (WIP) stored in it. The decision on the buffer capacities and their allocation is well known as the Buffer Allocation Problem (BAP).

The BAP is a well-researched problem which is hard to solve. On the one hand, the exact performance evaluation of flow lines is only possible for small systems under specific assumptions, and on the other hand, the allocation of buffer capacities is an NP-hard combinatorial problem (Smith and Cruz, 2005). Therefore, exact solutions for the BAP exist only for special cases (Enginarlar et al., 2005). However, heuristic search algorithms in combination with approximative evaluation methods are frequently used. The solution quality of these approaches is typically investigated numerically. Gershwin and Schor (2000) provide a comprehensive overview of solution approaches for the BAP published prior to the year 2000.

We provide a survey of the characteristics of the lines for analyzed instances of the BAP. We focus on unreliable serial lines with finite intermediate buffers and a single machine per station that processes discrete material (Figure 2.1). Further, we discuss different problem formulations of the BAP and their solution approaches. We include references that have been published after the review of Gershwin and Schor (2000).

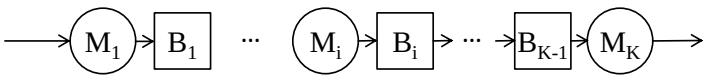



Figure 2.1.: Serial production line with $K$ stations (circles) and $K-1$ buffers of capacity $B_i$ (rectangles)

The remainder is organized as follows: Section 2.2 provides a classification of flow line characteristics. Section 2.3 addresses the different versions of the decision problem and the corresponding solution approaches. Concluding remarks and suggestions for future research are provided in Section 2.4.

## 2.2. Classification scheme for characteristics of flow lines

The key characteristics of serial lines are the number of stations, $K$, and the stations' stochastic properties. A station is characterized by the distribution of the processing times, the times to failure (TTF), and the times to repair (TTR). We found the following distributions in the literature: Deterministic (DET), Exponential (EXP), Erlang (ERL), Rayleigh (RA), Geometric (GEO), Uniform (U), Gamma (GAMMA), Normal (NORM), Lognormal (LOGN), and Bernoulli (BER). We distinguish whether all machines have the same (balanced line) or different properties (unbalanced line). We include references only if all of these key characteristics are clearly documented with published parameters for all distributions.

In addition to the key characteristics, a set of assumptions about the flow of workpieces in the line is required in order to reproduce the dynamics of a flow line (Dallery and Gershwin, 1992). An assumption has to be made on the supply of raw material in front of the first machine, which can be unlimited, i.e., saturated, or limited. Similarly, the demand for finished goods can be a limiting factor or there is a saturated demand. Moreover, the type of blocking has to be defined. If a buffer is full, the upstream station may either process an additional workpiece which then remains on the station until space in the downstream buffer becomes available, i.e., blocking after service (BAS), or no workpiece enters the machine until a buffer space becomes available, i.e., blocking before service (BBS). Unreliable stations can experience operation-dependent (OD) or time-dependent (TD) failures. In the former case, a station fails only while it is processing workpieces, while in the latter case, breakdowns occur independently of the operational status. If a failure occurs while a workpiece is being processed, it has to be specified whether the progress on the workpiece is conserved or lost. The differentiation becomes obsolete for exponentially distributed processing times or discrete-time models with Bernoulli and Geometric failures if the processing time equals the time interval length. In several cases these detailed assumptions are not reported on in the surveyed papers. We mark missing information by * and not applicable categories by - in the tables.

Notably, many references lack the required information to reproduce the instance of the line. Other features receive only little or no attention and are therefore not included in the table. For example scrap is only considered by Han and Park (2002). Moreover, correlations in the processing times are addressed only by Weiss and Stolletz (2015). They demonstrate that correlations can have a substantial impact on the optimal buffer allocation.

Table 2.1 shows unreliable lines reported in the literature after the review of Gershwin and Schor (2000). Two-thirds of the references consider flow lines that are balanced. Processing times are mostly deterministic with exponentially or geometrically distributed TTF and TTR. In almost all other cases processing times are exponentially or Erlang-distributed, again with exponentially distributed TTF and TTR. It can be observed that OD-failures dominate TD-failures. For the majority of the references the assumptions on conservation of work during failures is either not applicable or not addressed. With respect to the supply of the line, all but one of the articles assume unlimited supply. Lee and Ho (2002) assume random arrivals with exponentially distributed inter-arrival times. The blocking policy is often not defined. For the cases in which the blocking policy is defined, BBS occurs twice as often as BAS.

Some instances of flow lines are used by multiple authors. Kose and Kilincci (2015), Demir et al. (2011), Lee et al. (2009), and Nahas et al. (2006) use instances of Gershwin and Schor (2000). Instances proposed by Papadopoulos and Vidalis (2001) are utilized by Sabuncuoglu et al. (2006). Furthermore, Bekker (2013), Dolgui et al. (2007), Alon et al. (2005), and Dolgui et al. (2002) base their choice of instances on Vouros and Papadopoulos (1998).

## 2.3. Classification scheme for decision problems

The literature encompasses three main versions of the BAP. They all share the decision on the vector $\mathbf{B} = (B_1, B_2, ..., B_i, ..., B_{K-1})$, where $B_i$ represents the capacity of the buffer behind station $i$.

Table 2.1.: Characteristics of unreliable flow lines

| Reference | No. of stations | Processing time distr. | TTF distr. | TTR distr. | Unbalanced | Saturated supply | Saturated demand | Blocking type | Failure type | Work-conserving |
|---|---|---|---|---|---|---|---|---|---|---|
| Alon et al. (2005) | 3,5,6,10 | EXP | EXP | EXP | x | x | x | * | TD | - |
| | 5 | ERL | EXP | EXP | x | x | x | * | TD | * |
| Bekker (2013) | 5 | EXP | EXP | EXP | x | x | x | * | OD | - |
| | 5 | LOGN | EXP | EXP | x | x | x | * | OD | * |
| Chiang et al. (2000) | 15 | DET | EXP | EXP | x | x | x | BBS | OD | * |
| Demir et al. (2011) | 5,9,10,12,20,40 | DET | GEO | GEO | | x | x | * | * | - |
| Diamantidis and Papadopoulos (2004) | 4-6,10 | DET | BER | BER | x | x | x | * | OD | - |
| Dolgui et al. (2002) | 5 | DET | EXP | EXP | x | x | x | * | OD | * |
| Dolgui et al. (2007) | 5 | DET | EXP | EXP | x | x | x | * | OD | * |
| Enginarlar et al. (2002) | 2-20 | DET | EXP | EXP | | x | x | BBS | * | * |
| | 2-20 | DET | ERL | ERL | | x | x | BBS | * | * |
| | 2-20 | DET | RA | RA | | x | x | BBS | * | * |
| Enginarlar et al. (2005) | 3-30 | DET | EXP | EXP | | x | x | BBS | TD | * |
| Gershwin and Schor (2000) | 5,10,12,20,30 | DET | GEO | GEO | x | x | x | * | * | - |
| | 3,20 | DET | GEO | GEO | | x | x | * | * | - |
| | 7 | DET | EXP | EXP | x | x | x | * | * | * |
| Han and Park (2002) | 5,10 | DET | GEO | GEO | x | x | x | * | * | - |
| | 5,10 | DET | GEO | GEO | | x | x | * | * | - |
| Helber (2001) | 6 | DET | GEO | GEO | | x | x | * | OD | - |
| Kim and Lee (2001) | 3,8,10 | EXP | EXP | EXP | x | x | x | BAS | OD | x |
| Kose and Kilincci (2015) | 5,10 | DET | GEO | GEO | x | x | x | * | * | - |
| | 9,20,40 | DET | GEO | GEO | | x | x | * | * | - |
| Lee et al. (2009) | 5 | DET | GEO | GEO | x | x | x | BBS | OD | - |
| Lee and Ho (2002) | 5,6 | EXP | EXP | EXP | x | | x | * | * | - |
| | 5,6 | EXP | EXP | EXP | | | x | * | * | - |
| Li (2013) | 9,20 | DET | EXP | EXP | x | x | x | * | * | * |
| Massim et al. (2010) | 3,5,10 | DET | EXP | EXP | x | x | x | * | OD | * |
| Matta et al. (2012) | 5 | DET | EXP | EXP | x | x | x | * | OD | * |
| | 12 | DET | GEO | GEO | x | x | x | * | OD | - |
| Nahas et al. (2006) | 7 | DET | EXP | EXP | x | x | x | * | * | * |
| Papadopoulos and Vidalis (2001) | 3-6 | EXP | EXP | EXP | x | x | x | BAS | OD | x |
| Sabuncuoglu et al. (2006) | 3,5,10 | DET | EXP | EXP | | x | x | * | OD | x |
| | 4-6,8-10 | EXP | EXP | EXP | x | x | x | * | OD | x |
| | 4,5,7-10,12 | DET | EXP | EXP | x | x | x | * | OD | x |
| Savsar (2006) | 5 | EXP | EXP | U | | x | x | * | OD,TD | * |
| | 7 | DET | U/EXP/NORM/ ERL/GAMMA | U/NORM/ LOGN/DET | x | x | x | * | OD,TD | * |
| Shi and Gershwin (2009) | 3-6,12 | DET | GEO | GEO | x | x | x | * | OD | - |
| Shi and Gershwin (2014) | 30,70 | DET | GEO | GEO | | x | x | * | OD | - |
| | 20 | DET | GEO | GEO | x | x | x | * | OD | - |
| Shi and Men (2003) | 9 | DET | GEO | GEO | | x | x | * | * | - |
| Tempelmeier (2003) | 8,19,23 | DET | EXP | EXP | x | x | x | * | OD | * |
| | 14 | ERL | EXP | EXP | x | x | x | * | OD | * |
| | 14 | EXP | EXP | EXP | x | x | x | * | OD | - |
| Weiss and Stolletz (2015) | 14,24 | DET/ERL | EXP | EXP | x | x | x | BAS | OD | x |

(i) *Primal Problem:*

$$\min \sum_{i=1}^{K-1} B_i \tag{2.1a}$$

s.t.

$$\mathrm{E}[Th(\mathbf{B})] \geq Th^* \tag{2.1b}$$

$$B_i \in \mathbb{N}_0, \qquad 1 \leq i \leq K-1 \tag{2.1c}$$

The objective of the primal problem is to minimize the total buffer capacity in the line while ensuring that the expected throughput, $\mathrm{E}[Th(\mathbf{B})]$, equals or exceeds a given desired throughput, $Th^*$. $Th^*$ is usually selected as percentage of the theoretically achievable throughput in a line with infinite buffers.

(ii) *Dual Problem:*

$$\max \mathrm{E}[Th(\mathbf{B})] \tag{2.2a}$$

s.t.

$$\sum_{i=1}^{K-1} B_i = B_{tot} \tag{2.2b}$$

$$B_i \in \mathbb{N}_0, \qquad 1 \leq i \leq K-1 \tag{2.2c}$$

The dual problem with respect to the introduced primal (2.1) is the maximization of the expected throughput subject to the total buffer capacity, $B_{tot}$, available in the line. The value of $B_{tot}$ is usually given by space requirements on the shop floor. However, the dual problem may also be used to solve the primal problem by repetitively solving the dual for several values of total buffer capacities (Lee et al., 2009; Tempelmeier, 2003).

(iii) *Profit Problem:*

$$\max \text{Profit} = \alpha \mathrm{E}[Th(\mathbf{B})] - \beta \mathrm{E}[WIP(\mathbf{B})] - \gamma \sum_{i=1}^{K-1} B_i \tag{2.3a}$$

s.t.

$$\sum_{i=1}^{K-1} B_i \leq B_{tot} \tag{2.3b}$$

$$\mathrm{E}[Th(\mathbf{B})] \geq Th^* \tag{2.3c}$$

$$B_i \in \mathbb{N}_0, \qquad 1 \leq i \leq K-1 \tag{2.3d}$$

An attempt to directly balance the economic benefits of throughput with the buffer-related costs in the objective function is the profit problem. It uses weightings $\alpha$, $\beta$, and $\gamma$ to convert the technical measures of expected throughput, expected WIP, and buffer capacities into monetary units. The objective is to maximize the profit resulting from the gained revenue under the consideration of costs for the buffer capacities and the WIP stored in them. There is a constrained and an unconstrained version of the profit problem, i.e., Constraints (2.3b) and (2.3c) are not necessarily part of the decision problem. In the references considered, the parameters $\alpha, \beta$, and $\gamma$ are chosen without a direct link to empirical data.

(iv) *Other Problems:*
The works of Kim and Lee (2001) and Lee and Ho (2002) consider special cases of the BAP. Kim and Lee (2001) solely focus on the cost originating from the expected WIP, whereas Lee and Ho (2002) omit WIP-related costs and include costs for occurring throughput losses. Helber (2001) emphasizes that cash flows from revenue and investments in buffer capacities have different time scales. Thus, Helber (2001) suggests the use of a net present value approach. The problems introduced so far are all based on a single objective. Another idea is a multi-objective function. This approach delivers pareto-optimal solutions. Bekker (2013) employs this concept for the conflicting goals of throughput and WIP.

Table 2.2 lists the types of decision problems and the solution approaches that can be found in the literature. Most of the references address the primal or the dual problem. Both are addressed equally often. The minority of the references covers the optimization of profits.

The solution approaches for the BAP include a generative and an evaluative part. The generative method selects candidate solutions which have to be evaluated. The evaluation method determines the performance of the line, e.g., expected throughput or expected WIP, for a given buffer allocation. Sometimes integrated approaches are applied. Weiss and Stolletz (2015) use a Benders Decomposition approach which is based on a mixed integer programming formulation. In this special case, the corresponding master- and subproblem divide the approach into an integer programming-based generative and an evaluative method. An approach only delivers exact solutions if the generative and the evaluative part are both exact. Note that the simulation result converges to the exact solution if the length of the simulation run or the number of replications is chosen large enough. We therefore mark simulation with (x) in the table. Exact results for both, the generative and the evaluative method, are obtained only for two-machine lines (Enginarlar et al., 2002, 2005). For long sim-

ulation runs, Weiss and Stolletz (2015) also provide exact results. Metaheuristics, such as Genetic algorithms (GA), tabu search (TS), and artificial neural networks (ANN) are developed mainly for the dual problem. In contrast, rule-based allocation strategies and search algorithms are often employed for the primal problem. Maximization of profit functions is mainly addressed by genetic algorithms and gradient methods. Evaluation approaches are typically based on simulation, decomposition, and aggregation.

Table 2.2.: Characteristics of the decision problems

| | Decision Problem | | | | Solution Approach | | | |
|---|---|---|---|---|---|---|---|---|
| Reference | Primal | Dual | Profit | Others | Generative method | Exact | Evaluation method | Exact |
| Alon et al. (2005) | | x | | | Alias method based on cross entropy | | Simulation | (x) |
| Bekker (2013) | | | | x | Cross entropy method | | Simulation | (x) |
| Chiang et al. (2000) | x | | | | Rule of thumb | | Aggregation | |
| Demir et al. (2011) | | x | | | TS | | DDX | |
| | x | | | | Binary search and TS | | DDX | |
| Diamantidis and Papadopoulos (2004) | | x | | | Dynamic Programming | | Aggregation | |
| Dolgui et al. (2002) | | | x | | GA | | Aggregation | |
| Dolgui et al. (2007) | | | x | | Hybrid GA and Branch and Bound | | Aggregation | |
| Enginarlar et al. (2002) | x | | | | Analytical solution | x | Analytical solution | x |
| | x | | | | Buffer allocation rule | | - | |
| Enginarlar et al. (2005) | x | | | | Analytical solution | x | Analytical solution | x |
| | x | | | | Analytical solution | x | Aggregation | |
| | x | | | | Buffer allocation rule | | - | |
| Gershwin and Schor (2000) | x | | | | Search algorithm | | DDX/ADDX | |
| | | x | | | Gradient algorithm | | DDX/ADDX | |
| | | | x | | Gradient algorithm | | DDX/ADDX | |
| Han and Park (2002) | x | | | | Steepest descent with penalty function | | Aggregation | |
| Helber (2001) | | | | x | Gradient algorithm | | Decomposition | |
| Kim and Lee (2001) | | | | x | Local search | | Decomposition | |
| Kose and Kilincci (2015) | | x | | | Hybrid GA and Simulated Annealing | | Simulation | (x) |
| Lee et al. (2009) | | x | | | ANN and GA | | Simulation | (x) |
| Lee and Ho (2002) | | | | x | Modified responds surface methodology | | Simulation | (x) |
| Li (2013) | x | | | | Bottleneck-based iterative approach | | Approx. analytical formula | |
| Massim et al. (2010) | | | x | | Artificial immune algorithm | | DDX | |
| Matta et al. (2012) | x | | | | Numerical optimization technique | | Kriging approximation | |
| Nahas et al. (2006) | | x | | | Degraded ceiling approach | | ADDX | |
| Papadopoulos and Vidalis (2001) | | x | | | Sectioning approach | | Markovian state model | |
| Sabuncuoglu et al. (2006) | | x | | | Search algorithm | | Simulation | (x) |
| Savsar (2006) | | x | | | Enumeration | | Simulation | (x) |
| Shi and Gershwin (2009) | | | x | | Gradient method | | Decomposition | |
| Shi and Gershwin (2014) | | | x | | Gradient method with segmentation | | Decomposition | |
| Shi and Men (2003) | | x | | | Hybrid nested partition and TS | | DDX | |
| Tempelmeier (2003) | x | | | | Search algorithm and gradient-based search | | ADDX | |
| | | x | | | Gradient-based search | | ADDX | |
| Weiss and Stolletz (2015) | x | | | | Integer program | x | Simulation | (x) |

## 2.4. Conclusion and future research

We introduce a classification scheme that is used to describe existing unreliable flow lines for which the BAP is solved in its different problem formulations. Common assumptions are unlimited supply and an infinite last buffer. Failure type, conservation of work, and blocking type are only reported on insufficiently. Most of the references consider the primal and the dual problem. The maximization of a profit function is only considered in few cases. The corresponding solution approaches are mostly heuristic for both the generative and the evaluation part. Although some instances are used in several publications, there is a need for a library of sample instances with a complete description of the line characteristics and the allocations obtained with different solution approaches.

# 3. Buffer allocation using exact linear programming formulations and sampling approaches

*Co-author:*

**Raik Stolletz**
Chair of Production Management, Business School, University of Mannheim, Germany

*Published in:*

Manufacturing Modelling, Management, and Control, Volume 7(1), St. Petersburg, Russia, 2013, pages 1435-1440, DOI: 10.3182/20130619-3-RU-3018.00461

*Abstract:*

Several sampling approaches have been proposed in the literature for the analysis of flow lines with stochastic processing times and finite buffer capacities. The system's performance can be evaluated by a linear programming formulation if the capacities of the buffers between the stations are given. This work presents several mixed integer programming approaches to optimize the buffer allocation in flow lines with stochastic processing times. Sampling is used to represent the random processing times. The objective is to minimize the overall number of buffer spaces while obtaining at least a given goal production rate. Numerical experiments are carried out to evaluate different sampling approaches and model formulations. These approaches are compared regarding the robustness of the allocation decision with respect to the sample sizes.

## 3.1. Introduction

Flow lines are characterized by stochastic influences due to random processing times, machine breakdowns, and uncertain times of repair. This can lead to blocking and starvation of the stations in the line. Blocking occurs if a station $k$ finishes processing a workpiece and the downstream buffer is full. Therefore, the workpiece cannot depart from the station until a buffer space becomes available. This means that station $k$ cannot continue even if it is idle. In case of starvation, station $k$ finishes processing a workpiece but cannot continue processing because the upstream buffer is empty. Consequently, station $k$ idles until station $k-1$ finishes processing. Both effects have a strong impact on the line's performance. They cause a reduction of the theoretical throughput of the line. The allocation of additional buffer spaces can reduce these effects although it leads to an increase of the average work-in-process in the line.

Two basic streams of research can be found in the literature regarding the allocation of buffer capacities in stochastic flow lines, performance evaluation and optimization. The amount of literature on performance evaluation of flow lines is large. It can be classified into simulation, analytical exact methods, and analytical approximation methods. Discrete-event simulation (DES) is often used in the performance analysis due to its simplicity in terms of flow line modeling. Other papers propose exact numerical evaluation models. An overview on different approaches can be found in Gershwin and Schor (2000). In particular for large systems, approximation methods, e.g., decomposition, are an important tool for the evaluation of flow lines. Decomposition consists of dividing a flow line into smaller subsystems and recombining the subsystem solutions. The difficulty is the connection of these subsystems in such a way that the properties of the entire flow line are reflected sufficiently. Several decomposition approaches are pointed out in Dallery and Gershwin (1992).

The methods on performance evaluation proposed in the literature can also be used for optimization. However, a systematic optimization of flow line design using DES cannot be carried out efficiently because of long computation times. In contrast, due to the low computation times, a systematic optimization using exact analytical methods is possible, yet the underlying mathematical assumptions restrict their use in practice (Gershwin and Schor, 2000). A review on studies published after 1998 including optimization approaches can be found in Demir et al. (2014).

Dolgui et al. (2007) and Dolgui et al. (2013) prove NP-hardness for unreliable tandem production lines with oracle representation of revenue and costs functions as

objective and for series-parallel lines with stepwise revenue functions.
Recent approaches for the analysis and optimization of flow lines with limited buffer capacities are proposed by Matta and Chefson (2005) as well as Helber et al. (2011). Matta and Chefson (2005) use sensitivity analysis to determine the optimal allocation of buffer capacities based on a mathematical programming formulation originally proposed by Schruben (2000). Helber et al. (2011) introduce a discrete-time linear programming formulation which incorporates the buffer allocation problem. They transform the realizations of the stochastic processing times of the different jobs at a given production stage into corresponding realizations of production capacities. Besides the simulation error due to sampling, this method also leads to time discretization errors.
For the reasons mentioned above, we propose an exact mixed-integer programming (MIP) formulation which optimizes the number of buffer spaces behind each station using samples of the processing times in continuous time. In contrast to Helber et al. (2011), the optimization model is obtained by an exact linearization of the evaluation model without discretization errors.

## 3.2. Mixed-integer programming formulations

This section introduces several MIP formulations which optimize the overall number of buffer spaces assuming that a goal production rate is given.

### 3.2.1. Basic idea and assumptions

If the number of buffer spaces, $b_k$, behind each station $k$ is given, the performance of the line can be evaluated by a MIP formulation (Matta and Chefson, 2005). The key idea of this approach is to model the flow of a large number of workpieces throughout the line. Therefore, the start and finishing times of processing a workpiece $w$ at a station $k$ are represented by a set of real-valued decision variables. The random processing times are replaced by a deterministic sample. The sample generation will be discussed in the subsequent section in detail. The model of the flow line is based on the following assumptions:

- The material supply of the first station is unlimited. This means that 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. The MIP uses sampled processing times, $d_{k,w}$, for each station, $k$, and each workpiece, $w$.
- In case 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 already included in the processing times.

Figure 3.1 shows an example of a flow line according to these assumptions.

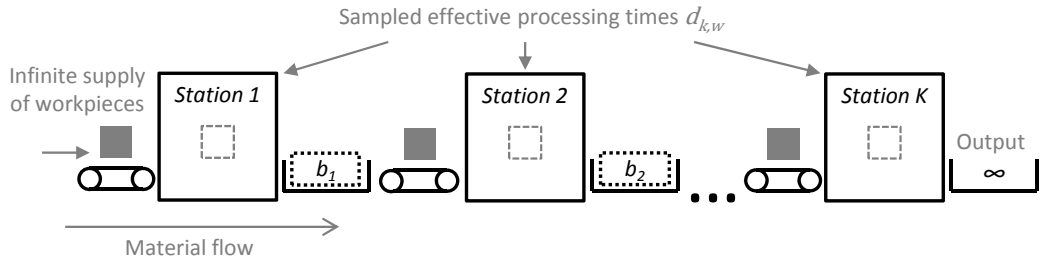



Figure 3.1.: Serial flow line with sampled processing times

### 3.2.2. MIP for the optimization

If the buffer capacities, $b_k$, are unknown, an optimization of the buffer capacities is necessary. This cannot be accomplished by the evaluation model introduced by Matta and Chefson (2005) as the buffer constraint becomes non-linear when the buffer capacity is a decision variable. Additional constraints on the performance of the line become necessary when the overall number of buffer spaces is optimized. Otherwise, the optimal solution would be a line without any buffer space. Therefore, a goal throughput has to be attained in steady-state. To consider steady-state performance measures, a warm-up of $W_0$ finished workpieces is excluded from the calculation of the throughput.
The formulation of the optimization model requires the definition of a set $b = 0, \ldots, B$ of possible buffer capacities as well as a binary variable $Y_{k,b}$. This binary variable indicates whether the number of buffer spaces behind station $k$ equals $b$. Based on the notation given in Table 3.1, the corresponding optimization model is presented in Formulation 1.

Table 3.1.: Notation for the MIP models

| | |
|---|---|
| **Sets and indices** | |
| $w = 1, \ldots, W$ | Workpieces |
| $k = 1, \ldots, K$ | Stations in the flow line |
| $b = 0, \ldots, B$ | Buffer capacities |
| **Parameters** | |
| $d_{k,w}$ | Processing time of workpiece $w$ at station $k$ |
| $M$ | Big-M |
| $TH^*$ | Goal throughput |
| $W_0$ | Number of workpieces in the warm-up phase |
| **Real-valued decision variables** | |
| $XS_{k,w}$ | Starting time of workpiece $w$ at station $k$ |
| $XF_{k,w}$ | Departure time of workpiece $w$ from station $k$ |
| $\overline{X}_k$ | Buffer capacity behind station $k$ |
| **Binary decision variables** | |
| $Y_{k,b} = \begin{cases} 1 \\ 0 \end{cases}$ | $\begin{array}{l} \text{if the buffer capacity behind station } k \text{ is equal to } b \\ \text{otherwise} \end{array}$ |

**Formulation 1**

$$\text{Minimize} \sum_{k=1}^{K-1} \overline{X}_k \tag{3.1}$$

s.t.

$$XS_{k,w} + d_{k,w} \leq XF_{k,w}, \qquad \forall k,\ \forall w \tag{3.2}$$

$$XF_{k,w} \leq XS_{k+1,w}, \qquad \forall k < K,\ \forall w \tag{3.3}$$

$$XF_{k,w} \leq XS_{k,w+1}, \qquad \forall k,\ \forall w < W \tag{3.4}$$

$$XF_{K,W} - XF_{K,W_0} \leq \frac{W - W_0}{TH^*} \tag{3.5}$$

$$XS_{k+1,w} - XF_{k,w+b} \leq M \cdot (1 - Y_{k,b}), \qquad \forall k < K, \forall b, \forall w \leq W - b \tag{3.6}$$

$$\sum_{b=0}^{B} Y_{k,b} = 1, \qquad \forall k < K \tag{3.7}$$

$$\overline{X}_k = \sum_{b=0}^{B} b \cdot Y_{k,b}, \qquad \forall k < K \tag{3.8}$$

$$Y_{k,b} \in \{0,1\}, \qquad \forall k < K, \forall b \tag{3.9}$$

$$XS_{k,w}, XF_{k,w} \geq 0, \qquad \forall k, \forall w \tag{3.10}$$

The objective function (3.1) minimizes the overall number of buffer spaces in the line. Constraints (3.2) state that a workpiece, $w$, departs from station $k$ at the earliest after being processed. Hereby, the slack variable of the inequality corresponds to the blocking time of workpiece $w$ after being processed at station $k$. A workpiece cannot start processing at station $k+1$ until it finishes processing at station $k$. This is ensured by Equations (3.3). The slack variable of this inequality defines the waiting time of workpiece $w$ in the buffer between station $k$ and station $k+1$. As a station can only process one workpiece at a time, Equations (3.4) state that workpiece $w+1$ does not start processing at station $k$ before workpiece $w$ departs from this station. A station may starve between the processing of two consecutive workpieces, which is determined by the slack related to Equations (3.4). Equation (3.5) ensures that a goal throughput, $TH^*$, is attained. The realized throughput is calculated by the fraction of the number of finished parts, $(W - W_0)$, and the required time, $(XF_{K,W} - XF_{K,W_0})$, in steady-state, i.e., after the warm-up phase. Constraints (3.6) ensure that the buffer capacity is not exceeded. The inequality is valid in the event of a buffer capacity of $b = \overline{X}_k$ because the right-hand side (RHS) becomes zero. In this case, the inequality ensures that workpiece $w$ departs from the buffer between station $k$ and station $k+1$ before workpiece $w+b$ enters. For $b \neq \overline{X}_k$, the equation becomes redundant due to the Big-M on the RHS. If there is no buffer between station $k$ and station $k+1$, i.e., $b = 0$, the inequality reduces to $XS_{k+1,w} \leq XF_{k,w}$. Together with Equations (3.3) it is ensured that the time when workpiece $w$ departs from station $k$ equals the starting time of $w$ at station $k+1$. The capacity of each buffer between two stations has to be unique. This is stated in Equations (3.7). Constraints (3.8) connect the (redundant) buffer space variable, $\overline{X}_k$, and the binary variables $Y_{k,b}$.

This formulation is similar to the one presented by Matta (2008), with the exception of the assumption of blocking after service.

### 3.2.3. Other MIP formulations

Instead of modeling the binary buffer capacity indicator variable as presented in Section 3.2.2, a binary variable $X_{k,b}$ can be used, which is defined as

$$X_{k,b} = \begin{cases} 1 & \text{if the buffer capacity behind station } k \text{ is greater than} \\ & \text{or equal to } b \\ 0 & \text{otherwise.} \end{cases}$$

To obtain Formulation 2, Equations (3.1) to (3.5) and (3.10) remain as in Formulation 1. Equations (3.6) to (3.9) have to be replaced by Constraints (3.11) to (3.14).

**Formulation 2**

Equations (3.1) to (3.5) and (3.10)

$$XS_{k+1,w} - XF_{k,w+b} \leq M \cdot (1 - X_{k,b+1}), \qquad \forall k < K, \; \forall b < B, \; \forall w \leq W - b \tag{3.11}$$

$$\overline{X}_k \geq b \cdot X_{k,b}, \qquad \forall k < K, \; \forall b \tag{3.12}$$

$$X_{k,b} \in \{0, 1\}, \qquad \forall k < K, \; \forall b \tag{3.13}$$

$$\overline{X}_k \geq 0, \text{ integer}, \qquad \forall k < K \tag{3.14}$$

Note that in this case $\overline{X}_k$ does not necessarily assume integer values. Therefore, this has to be stated explicitly.

Formulations 3 and 4 are obtained by adding the redundant Equations (3.15) and (3.16) to Formulation 2 respectively.

**Formulation 3**

Equations (3.1) to (3.5) and (3.10) to (3.14)

$$X_{k,b} \geq X_{k,b+1}, \qquad \forall k < K, \; \forall b < B \tag{3.15}$$

**Formulation 4**

Equations (3.1) to (3.5) and (3.10) to (3.14)

$$\overline{X}_k \geq \sum_{b=1}^{B} X_{k,b}, \qquad \forall k < K \tag{3.16}$$

Equations (3.12) in Formulation 2 can be replaced by adding both Equations (3.15) and (3.16). This leads to Formulation 5.

**Formulation 5**

Equations (3.1) to (3.5), (3.10), (3.11), and (3.13) to (3.16)

Finally, Equations (3.12) can also be replaced by Equations (3.17), which leads to Formulation 6.

**Formulation 6**

Equations (3.1) to (3.5), (3.10), (3.11), (3.13), and (3.14)

$$\overline{X}_k = \sum_{b=1}^{B} b \cdot (X_{k,b} - X_{k,b+1}) + B \cdot X_{k,B}, \qquad \forall k < K \tag{3.17}$$

### 3.2.4. Sampling of effective processing times

Several sampling approaches are proposed in the literature. In this section, two of these approaches are discussed: Simple Random Sampling (SRS) and Descriptive Sampling (DS). SRS is the standard sampling procedure in a Monte Carlo Simulation, while DS was proposed by Saliby (1990a).

Both methods can be characterized by the division of a sample into two parts, the set and the sequence. The set is defined by all the values that occur in the sample independently of their particular order in this sample, i.e., the values in the set are sorted in ascending order. The sequence describes the particular order in which the set values occur in the sample.

The following example illustrates the division of a random sample into a set and a sequence (Saliby, 1990a). The sample is given by

$$\text{Sample} = \{0.45, 0.12, 0.63, 0.23, 0.84\}.$$

Then the corresponding sorted set is defined by the sample values in ascending order

$$\text{Set} = \{0.12, 0.23, 0.45, 0.63, 0.84\}$$

whereas the sequence defines the order of the set values in the sample

$$\text{Sequence} = \{3, 1, 4, 2, 5\}.$$

SRS generates a sample value by randomly selecting a value $r \in (0, 1)$ and transforming it with regard to a given distribution using the inverse transformation method. The sample is determined by iteratively repeating this step. In terms of set and sequence, this method is based on a random set and a random sequence, which are determined in one step. The use of a random set is usually justified by the argument that a sample has to be generated completely at random to represent random behavior. However, in a Monte Carlo Simulation it is assumed that the samples are following a given distribution. This implies that the random behavior is to be described according to this distribution.

Therefore, DS is based on a deterministic set and a random sequence. The usage of a deterministic set of values as an input for the sample is motivated by the fact that a random set causes a high variability of the results. Because of the deterministic set, DS leads to a more precise description of the sampled distribution (Saliby, 1990b). The randomness of the sample is only represented by the random permutation of the deterministic set values. Consequently, the generation of a descriptive sample requires two steps, the generation of the deterministic set and the random permutation of those values.

The variability of the simulation estimates depends on the variability of the input sample. Therefore, the variability of the estimates is influenced by both the set and the sequence of the sample. In a numerical study, Saliby (1990b) demonstrates that the set has a high impact on the estimates' variability. However, the set relative contribution is nearly constant, irrespective of the sample size. This means that a larger sample does not lead to an improvement regarding the set variability. Using DS, the sampling cumulative distribution function is close to the probability distribution function and thereby minimizes the set variability.

The generation of the deterministic set values requires prior knowledge of the input sample size $n$. An approximated value for the sample size is also sufficient. If the current sample size is underestimated, the residual values are drawn without replacement from the same set. If the current sample size is overestimated, a subset of the estimated sample set values is used.

Following a given distribution with distribution function $F(x)$, the set values are calculated by

$$xd_i = F^{-1}\left[\frac{(i-0.5)}{n}\right], \qquad i = 1,\ldots,n.$$

Note that the same set values are used for all replicated runs of an experiment as the set is deterministic.

In the second step these values are shuffled according to Algorithm 1, which is proposed by Saliby (1990a). This step has to be repeated for each replication, while it is sufficient to generate the set values only once for all replicated runs in an experiment.

**Input:** Sample size $n$, set values $xd_i$, index $i_p$
**Output:** Shuffled set values $xd_i$
Set $i_p = 1$
**while** $i_p < n$ **do**
Randomly generate integer $i_{aux} \in [i_p, n]$
Interchange $xd[i_p]$ with $xd[i_{aux}]$
Set $i_p = i_p + 1$;
**end while**

Algorithm 1: Random permutation of set values

## 3.3. Numerical results

The numerical study evaluates the following aspects:

1. Performance of the different MIP formulations
2. Impact of the sample size
3. Performance of the sampling approaches

In all examples, we consider an unbalanced flow line with $K = 3$ stations. The processing times at all stations are exponentially distributed. The first two stations operate with service rate $\mu_1 = \mu_2 = 7$ and the last station with $\mu_3 = 6$. The number of workpieces in the warm-up phase is selected as $W_0 = 2,000$. A goal throughput of $TH^* = 5.776$ has to be attained. The simulation runs have been performed on an Intel Core i5-2520M with 2.5 GHz and 4 GB RAM. The models are implemented in C++ using Gurobi 5.0.

The first experiment compares the computational effort of the optimization models presented in Sections 3.2.2 and 3.2.3. Table 3.2 presents the average computation

Table 3.2.: Average computation time (sec.) of different MIP formulations

| | MIP formulation | | | | | |
|---|---|---|---|---|---|---|
| W | 1 | 2 | 3 | 4 | 5 | 6 |
| 4,000 | 102.1 | 114.5 | 112.2 | 114.2 | 110.1 | 113.0 |
| 5,000 | 158.7 | 178.8 | 179.9 | 185.5 | 183.6 | 184.3 |
| 6,000 | 230.4 | 286.2 | 286.1 | 297.8 | 290.0 | 287.4 |
| 7,000 | 305.1 | 378.9 | 380.0 | 412.2 | 659.3 | 386.2 |
| 8,000 | 451.8 | 577.4 | 617.6 | 620.1 | 566.4 | 606.8 |
| 10,000 | 712.3 | 1,037.8 | 1,033.1 | 1,120.9 | 1,214.0 | 1,075.4 |

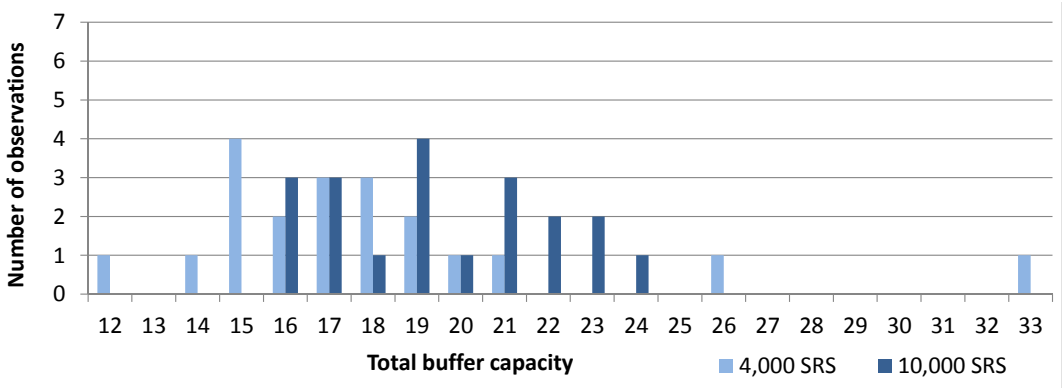



Figure 3.2.: Range of total buffer capacity for $W = 4,000$ and $W = 10,000$

times of different MIP formulations for varying number of workpieces based on 5 different samples. In all cases, Formulation 1 results in the lowest average computation times. The gap in computation time between the formulations rises with increasing number of workpieces. For instances with 4,000 workpieces, the difference in computation time is just a few seconds. In contrast, for instances with 10,000 workpieces, it is more than 5 minutes. Therefore, the subsequent experiments use Formulation 1.

The aim of the second experiment is to investigate the impact of the sample size on the robustness of the results. Several runs with different sample sizes are compared. Figure 3.2 displays the number of observations of the total buffer capacities obtained by 20 different samples generated by SRS with 4,000 and 10,000 workpieces respectively. The range of the total buffer capacity for a sample size of 4,000 workpieces is much larger than for a sample size of 10,000. A total buffer capacity between 12 and 33 is obtained for a sample size of 4,000, while the range lies between 16 and 24 for 10,000 workpieces. Consequently, the range of the total buffer capacity decreases with increasing sample size. Thus, regarding the robustness of

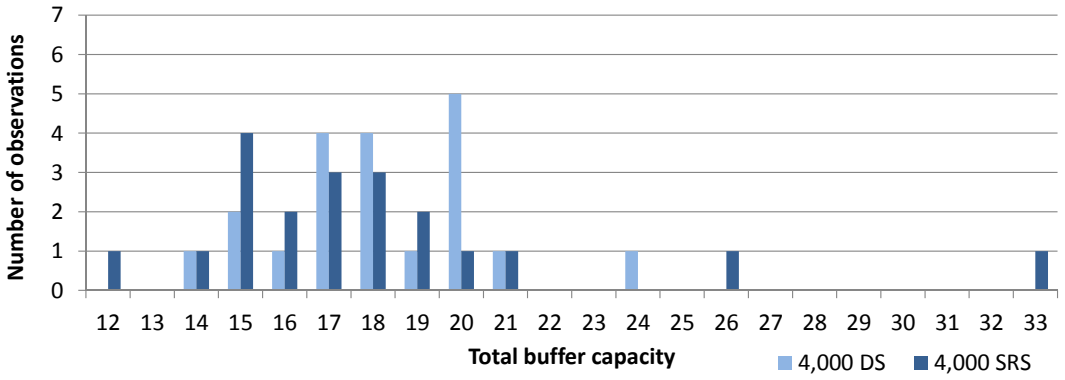



Figure 3.3.: Range of total buffer capacity for DS and SRS with $W = 4,000$

the solution, a larger sample size yields better results. Nevertheless, a sample size of 10,000 workpieces is still not sufficient. However, increasing sample sizes lead to increasing computation times. A complete run with 4,000 workpieces takes on average 137 seconds in our experiment, while a run with 10,000 workpieces takes 726 seconds on average.

The third experiment is carried out to analyze the performance of the different sampling methods, DS and SRS, introduced in Section 3.2.4. Figures 3.3 and 3.4 compare the range of the total buffer capacity of both methods for an amount of 4,000 and 10,000 workpieces respectively. Both figures show that the range of total buffer capacity is smaller using DS. SRS leads to a range from 12 to 33 overall buffer spaces in the case of 4,000 workpieces and a range of 16 to 24 overall buffer spaces in the case of 10,000 workpieces. In contrast, DS returns a range from 14 to 24 and a range of 16 to 22 overall buffer spaces in the case of 4,000 and 10,000 workpieces respectively. This shows that the robustness is higher for samples generated by DS since the range of the buffer capacities and hence the spread is smaller in both cases.

One of the goals of the numerical study is to investigate the optimal sample size. The experiments demonstrate that a sample of 10,000 workpieces is not sufficiently large even if DS is applied. However, a sample size of 10,000 workpieces already needs a computation time of 10 minutes using Formulation 1 and the computation time for larger samples turns out to be much longer. Using 20 samples of 100,000 workpieces each results in a smaller total buffer capacity of 18 or 19. The computation time adds up to more than 204 hours on average. In contrast to this, for a sample of 1,000,000 workpieces, the computation time takes more than 500 hours, again with 18 or 19 buffer spaces in total.

For the different combinations of a total buffer capacity of 18 and 19, which are obtained using samples of 10,000 workpieces, the throughput is evaluated with five

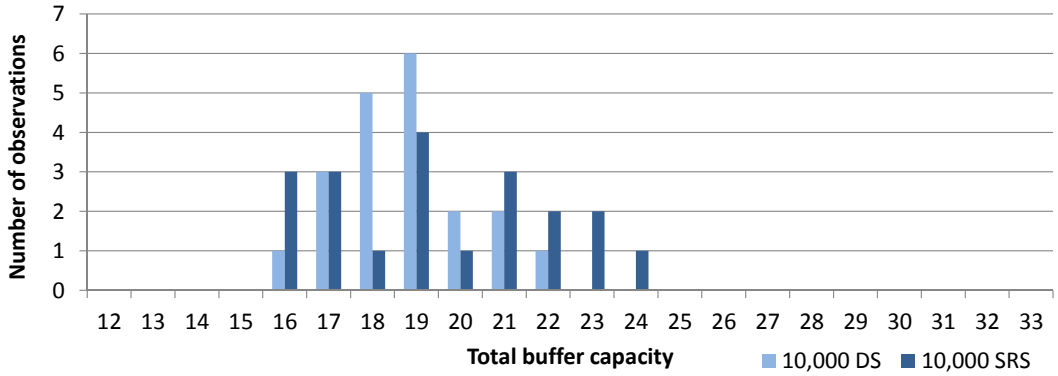



Figure 3.4.: Range of total buffer capacity for DS and SRS with $W = 10,000$

new samples of 1,000,000 workpieces each. Figure 3.5 presents the minimum, the average, and the maximum throughput that is obtained from these five samples for each buffer allocation. The goal throughput is always attained for the allocations with 19 buffer spaces in total. In contrast, in most of the cases, a buffer capacity of 18 is not sufficient to attain the goal throughput. The maximum deviation from the goal throughput in case of $X_1 = 6$ and $X_2 = 12$ equals 0.16%.

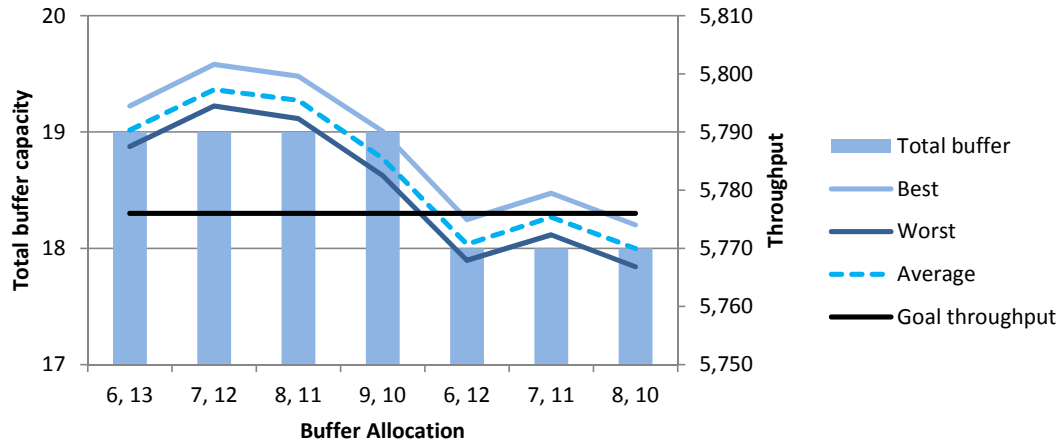



Figure 3.5.: Throughput evaluation

## 3.4. Conclusion

We introduce several MIP formulations for the optimization of the buffer allocation in stochastic flow lines. The stochastic processing times are modeled by samples. A numerical study is conducted to investigate the performance of these approaches and the robustness of the results. The numerical study demonstrates that MIP Formulation 1 can be solved faster than Formulations 2-6. Additionally, it is shown that DS leads to better results than SRS. Moreover, the larger the sample size, the more robust are the results for both sampling approaches. However, larger sample sizes lead to long computation times. A sample of 10,000 workpieces is hardly enough to obtain robust results but already takes around 10 minutes of computation time. As the computation time is still too long, it may be useful to investigate problem-specific optimization algorithms and heuristics.
In a further step, we would like to extend the model to other, more complex production systems such as flow lines with closed loops.

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

*Co-authors:*

**Raik Stolletz**
Chair of Production Management, Business School, University of Mannheim, Germany

*Published in:*

OR Spectrum, 2015, Volume 37(4), pages 869-902, DOI: 10.1007/s00291-015-0393-z

*Abstract:*

The allocation of buffer spaces 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.

## 4.1. Introduction

Flow lines consist of a number of stations that are arranged in series and separated by buffers with limited capacities. The workpieces flow through the system from station to station, waiting in the buffer if the downstream station is not available. This type of 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 to 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 the literature, is NP-hard (Smith and Cruz, 2005). Three types of objective functions can be found: minimization of the total buffer capacity with respect to a given goal throughput, throughput maximization with respect to a limited number of buffer spaces, and profit maximization. This paper focuses on the minimization of the total buffer capacity.

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 (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 station 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 the 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 the 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 4.2 introduces the MIP formulation for the optimization of flow lines. In Section 4.3, the Benders Decomposition approach and a procedure to obtain initial bounds are presented. Section 4.4 provides a numerical study on the performance of Benders Decomposition and the initial bounds. Section 4.5 presents the conclusions and further research efforts.

## 4.2. 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 Section 4.2.1. The Benders Decomposition approach is based on iterative generation of candidate allocations and evaluation of these candidates. Therefore, Section 4.2.2 presents a fast simulation algorithm for the throughput evaluation of a given buffer allocation. Finally, Section 4.2.3 describes the MIP 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 depar-

ture 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 Section 4.4.1 supports this claim for 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.

### 4.2.1. 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.
- A goal throughput rate of $TH^*$ has to be attained after the warm-up.

Figure 4.1 shows an example of a flow line according to these assumptions.

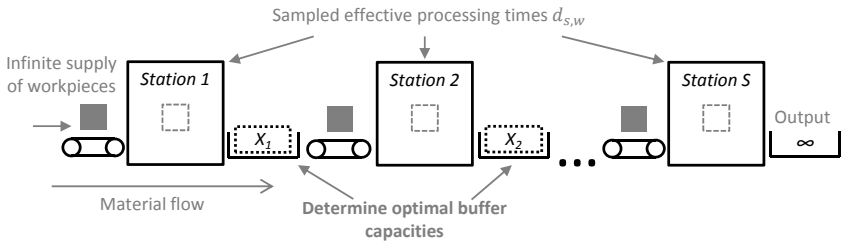



Figure 4.1.: Flow line under consideration

### 4.2.2. 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 2. The corresponding notation can be found in Table 4.1.

Table 4.1.: 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 $s$ |
| $TH^*$ | Goal throughput |
| $W_0$ | Number of workpieces in the warm-up phase |
| $M$ | Big-M (sufficiently large positive number) |
| **Real-valued decision variables** | |
| $XS_{s,w}$ | Start time of workpiece $w$ at station $s$ |
| $XF_{s,w}$ | Departure time of workpiece $w$ from station $s$ |
| $X_s$ | Buffer capacity behind station $s$ |
| **Binary decision variables** | |
| $Y_{s,b} = \begin{cases} 1 \\ 0 \end{cases}$ | $\begin{cases} \text{If the buffer capacity behind station } s \text{ is equal to } b \\ \text{Otherwise} \end{cases}$ |

1: $XS_{1,1} = 0$
2: **for** all stations $s < S$ **do**
3: $\quad XF_{s,1} = XS_{s,1} + d_{s,1}$
4: $\quad XS_{s+1,1} = XF_{s,1}$
5: **end for**
6: $XF_{S,1} = XS_{S,1} + d_{S,1}$
7: **for** all workpieces $w > 1$ **do**
8: $\quad$ **for** all stations $s < S$ **do**
9: $\quad\quad$ **if** $s = 1$ **then**
10: $\quad\quad\quad XS_{s,w} = XF_{s,w-1}$
11: $\quad\quad$ **else**
12: $\quad\quad\quad XS_{s,w} = \max\{XF_{s,w-1}, XF_{s-1,w}\}$
13: $\quad\quad$ **end if**
14: $\quad\quad$ **if** $X_s = 0$ **then**
15: $\quad\quad\quad XF_{s,w} = \max\{XS_{s,w} + d_{s,w}, XF_{s+1,w-1}\}$
16: $\quad\quad$ **else if** $X_s \geq w$ **then**
17: $\quad\quad\quad XF_{s,w} = XS_{s,w} + d_{s,w}$
18: $\quad\quad$ **else**
19: $\quad\quad\quad XF_{s,w} = \max\{XS_{s,w} + d_{s,w}, XS_{s+1,w-X_s}\}$
20: $\quad\quad$ **end if**
21: $\quad$ **end for**
22: $\quad XS_{S,w} = \max\{XF_{S,w-1}, XF_{S-1,w}\}$
23: $\quad XF_{S,w} = XS_{S,w} + d_{S,w}$
24: **end for**

Algorithm 2: Throughput calculation

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 has 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 $XS_{s,w}$ 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 workpiece is available. In this case, $s$ idles until station $s-1$ provides a workpiece (lines 12 and 22). Departure times $XF_{s,w}$ 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 $XF_{S,W} - XF_{S,W_0}$ after the warm-up phase:

$$TH = \frac{W - W_0}{XF_{S,W} - XF_{S,W_0}} \tag{4.1}$$

### 4.2.3. Optimization of buffer allocations

The problem of allocating a minimum number of total buffer spaces while achieving a given goal throughput can be solved by a MIP formulation as follows. Additionally to the notation used in Section 4.2.2, a binary variable $Y_{s,b}$ is required to indicate that the buffer capacity behind station $s$ equals $b$ (see Table 4.1).

$$\text{Minimize} \sum_{s=1}^{S-1} X_s \tag{4.2}$$

s.t.

$$XS_{s,w} + d_{s,w} \leq XF_{s,w}, \qquad \forall s,\ \forall w \tag{4.3}$$

$$XF_{s,w} \leq XS_{s+1,w}, \qquad \forall s \leq S-1,\ \forall w \tag{4.4}$$

$$XF_{s,w} \leq XS_{s,w+1}, \qquad \forall s,\ \forall w \leq W-1 \tag{4.5}$$

$$XF_{S,W} - XF_{S,W_0} \leq \frac{W - W_0}{TH^*} \tag{4.6}$$

$$XS_{s+1,w} - XF_{s,w+b} \leq M \cdot (1 - Y_{s,b}), \qquad \forall s \leq S-1\ \forall b,\ \forall w \leq W-b \tag{4.7}$$

$$\sum_{b=0}^{B_s} Y_{s,b} = 1, \qquad \forall s \leq S-1 \tag{4.8}$$

$$X_s = \sum_{b=0}^{B_s} b \cdot Y_{s,b}, \qquad \forall s \leq S-1 \tag{4.9}$$

$$XS_{s,w}, XF_{s,w} \geq 0, \qquad \forall s,\ \forall w \tag{4.10}$$

$$Y_{s,b} \in \{0,1\}, \qquad \forall s \leq S-1,\ \forall b \tag{4.11}$$

The objective function (4.2) minimizes the overall number of buffer spaces in the line. The constraints are linearizations of the formulas given in Algorithm 2. Constraint (4.3) 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 (4.4). 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 (4.5) 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 (4.5). Inequality (4.6) ensures that a goal throughput, $TH^*$, is attained (see Equality (4.1)). Constraint (4.7) 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 Big-M on the right-hand side (RHS). We choose Big-M 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 $XS_{s+1,w} \leq XF_{s,w}$. Together with Inequality (4.4), 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 Equation (4.8). Constraint (4.9) 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 (4.4) and (4.5) determines the start times as in Algorithm 2. Accordingly, the combination of Equations (4.3) and (4.7) determines the completion times.

If the buffer capacities behind each station are given, the MIP can also be used for evaluation (instead of Algorithm 2). 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 (4.2).

## 4.3. 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.

### 4.3.1. Adjustments and specific features

Figure 4.2 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 Section 4.2.3, the binary variables, $Y_{s,b}$, and the integer variables, $X_s$, become part of the master problem. The real-valued decision variables, $XS_{s,w}$ and $XF_{s,w}$, belong to the subproblem.

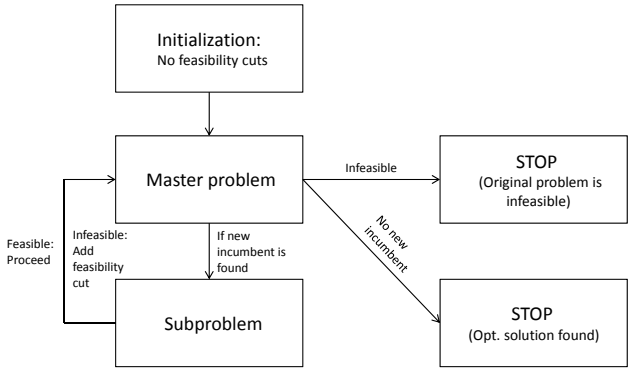



Figure 4.2.: Overview of Benders Decomposition for the BAP

Constraints (4.3)-(4.6) and (4.10) only contain real-valued decision variables and thus belong to the subproblem. Constraints (4.8), (4.9), and (4.11) are included in the master problem, as they only contain binary variables. Constraint (4.7) contains both types of variables. It forms part of the subproblem and contains the master variables, $X_s$, as parameters. Consequently, Constraint (4.7) can be replaced by (4.12).

$$XS_{s+1,w} - XF_{s,w+X_s} \leq 0, \quad \forall s \leq S-1, \forall w \leq W - X_s \tag{4.12}$$

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 (4.2) includes no real-valued decision variables. Thus, the objective function of the master problem is equal to the objective function (4.2). To avoid an overestimation of the throughput as outlined in Section 4.2.3, we use Algorithm 2 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 Equations (4.2), (4.8), (4.9), and (4.11), the complete master problem is defined as follows.

$$\text{Minimize} \sum_{s=1}^{S-1} X_s \tag{4.2}$$

$$\text{s.t.} \quad \sum_{b=0}^{B_s} Y_{s,b} = 1, \qquad \forall s \leq S-1 \tag{4.8}$$

$$X_s = \sum_{b=0}^{B_s} b \cdot Y_{s,b}, \qquad \forall s \leq S-1 \tag{4.9}$$

Feasibility Cuts

$$Y_{s,b} \in \{0,1\}, \qquad \forall s \leq S-1, \forall b \tag{4.11}$$

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 (4.13) (classical feasibility cut).

$$0 \geq -(\sum_{s=1}^{S-1}\sum_{w=1}^{W-b_s} \mu^h_{5,s,w,b_s} \cdot M \cdot (1 - Y_{s,b_s}) + \mu^h_4 \cdot \frac{W - W_0}{TH^*} - \sum_{s=1}^{S}\sum_{w=1}^{W} \mu^h_{1,s,w} \cdot d_{s,w}) \tag{4.13}$$

$\mu^h$ 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 Big-M coefficients in Constraints (4.7), the classical feasibility cuts (4.13) 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 Big-M 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 attained. 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 $b_s$:

$$1 \leq \sum_{s=1}^{S-1}\sum_{b=b_s+1}^{B_s} Y_{s,b}. \tag{4.14}$$

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.

### 4.3.2. Generation of lower bounds from subsystems

Figure 4.3 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 attains the goal throughput, the 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.

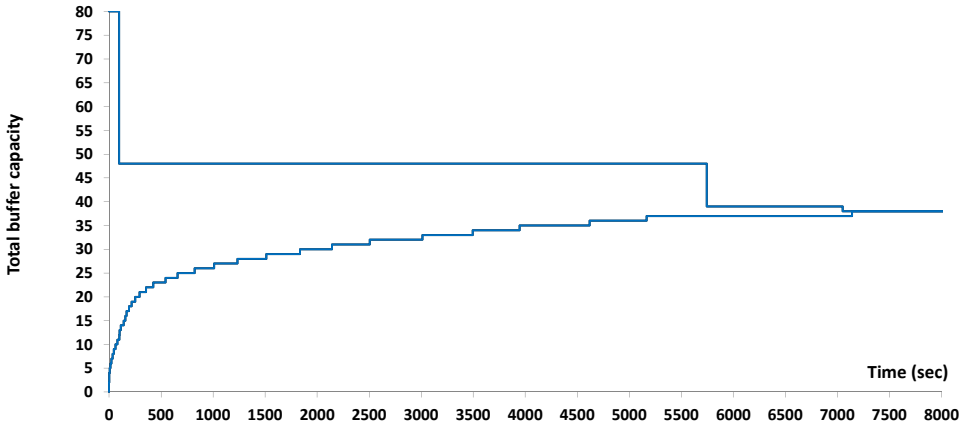



Figure 4.3.: Course of the lower and upper bounds during the solution process

In the 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. The 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 Section 4.4.

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 4.1.** *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 $Arr_w \geq 0$ be the arrival time of workpiece $w$ in the system with limited supply. According to Algorithm 2, the start time of workpieces 1 and 2 at the first station of the system with limited supply are calculated from $XS^{lim}_{1,1} = Arr_1$ and $XS^{lim}_{1,2} = \max\{XF^{lim}_{1,1}, Arr_2\} = \max\{XS^{lim}_{1,1} + d_{1,1}, Arr_2\}$ respectively. As $Arr_w = 0$ for all $w$ in the system with unlimited supply, the start times equal $XS^{unl}_{1,1} = 0$ and $XS^{unl}_{1,2} = XF^{unl}_{1,1} = XS^{unl}_{1,1} + d_{1,1}$. Consequently, $XS^{unl}_{1,2} \leq XS^{lim}_{1,2}$. With mathematical induction using the above formulas for $w$ and the formulas of Algorithm 2 to calculate start and departure times, it follows that $XF^{unl}_{S,W} \leq XF^{lim}_{S,W}$, i.e., less time to produce $W$ workpieces is required in the unlimited case, and therefore, the throughput of the system with unlimited supply is higher or equal to the throughput of an identical system with limited supply. □

**Theorem 4.2.** *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 $Dep_w \geq 0$ be the time workpiece $w$ is allowed to leave the system with limited supply. According to Algorithm 2, the departure time of workpiece 1 at the last station, $S$, is calculated as $XF^{lim}_{S,1} = \max\{XS_{S,1} + d_{S,1}, Dep_1\}$ for the system with limited outflow. As $Dep_w = 0$ for all $w$ in the system with unlimited outflow, the departure time for the system with limited outflow equals $XF^{unl}_{S,1} = XS_{S,1} + d_{S,1}$. Consequently, $XF^{unl}_{S,1} \leq XF^{lim}_{S,1}$. With mathematical induction using the above formulas for $w$ and the formulas of Algorithm 2 to calculate start and departure times, it follows that $XF^{unl}_{S,W} \leq XF^{lim}_{S,W}$, 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

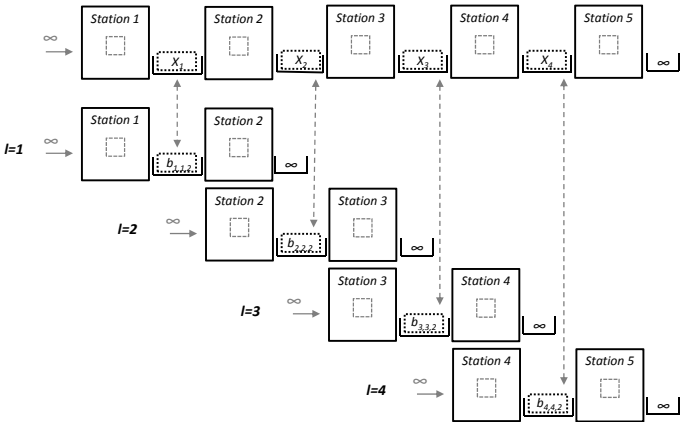



Figure 4.4.: Generation of lower bounds via subsystems of size $i = 2$

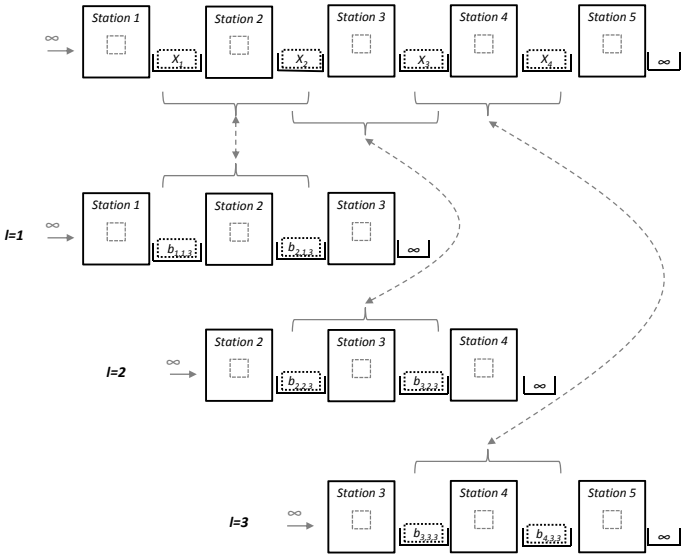



Figure 4.5.: Generation of lower bounds via subsystems of size $i = 3$

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 Figure 4.4. 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 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 $b_{s,l,i}$. Figure 4.5 depicts a line with 5 stations divided into subsystems of size $i = 3$.

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 4.5 illustrates this case for a subsystem of size $i = 3$. Inequality (4.15) presents the bounds obtained from subsystems of size $i \geq 3$.

$$\sum_{j=0}^{i-2} b_{j+l,l,i} \leq \sum_{j=0}^{i-2} X_{j+l} \qquad \forall l \tag{4.15}$$

We apply Benders Decomposition to solve each subsystem. The size of the subsystems is increased iteratively, until the size of the original line is attained. This procedure is depicted in Figure 4.6.

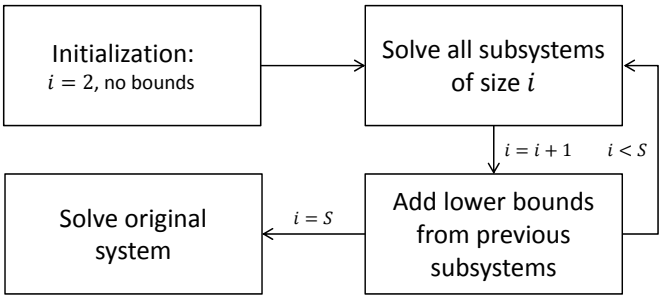



Figure 4.6.: Overview of bound calculation

## 4.4. Numerical study

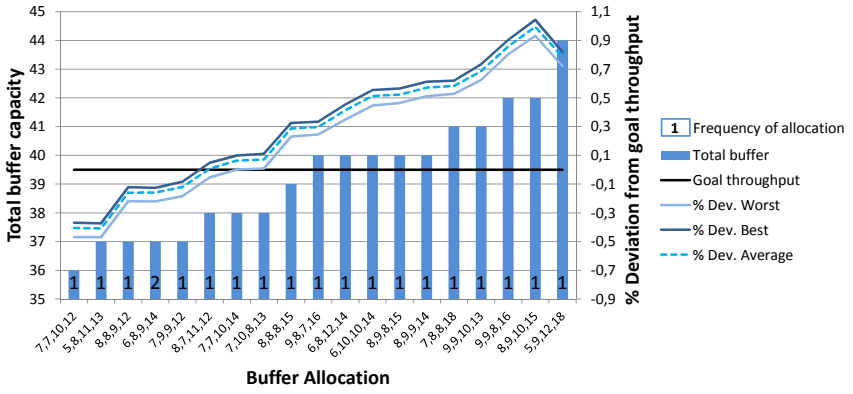
All of the algorithms are implemented in C++. Gurobi 5.0, with default settings, is used to solve the linear and mixed-integer programs. The numerical study is performed on an Intel Core i7-3930K with 6x 3.2 GHz and 32 GB RAM.
For all instances, the capacity of each buffer is limited to $B_s = 20$, and the warm-up phase is selected as $W_0 = 2,000$.
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 (4.13) or (4.14) 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.

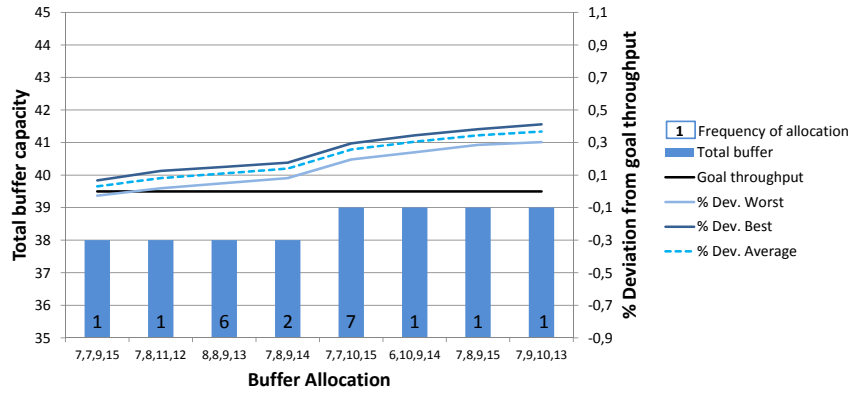
### 4.4.1. A note on robustness

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.

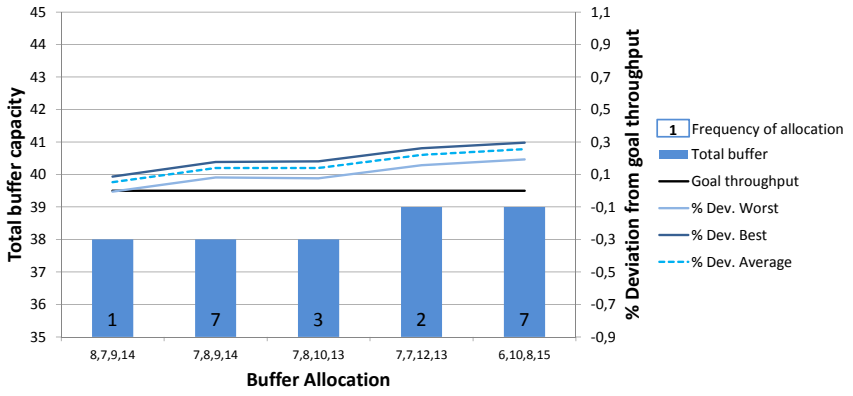
Figure 4.7 depicts the results of a throughput evaluation for different optimal buffer allocations for a varying number of workpieces. These allocations are obtained by independently solving 20 samples with 10,000 (Figure 4.7a), 250,000 (Figure 4.7b), 1,000,000 (Figure 4.7c), and 5,000,000 workpieces (Figure 4.7d) each. The throughput evaluation is conducted with 20 additional samples of 5,000,000 workpieces. Figure 4.7 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 (Figure 4.7a), 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 attained, whereas a total number of 37 (or less) is not (even in the best case) sufficient. On average, the goal throughput is attained 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 Figure 4.7b (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 attained for all allocations. Even in the worst case, the maximum deviation from the goal throughput equals 0.03%. Figure 4.7c 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 are 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



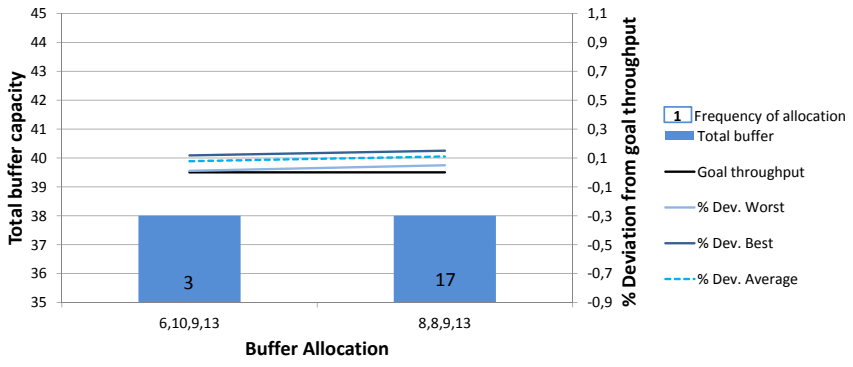

(a) $W = 10,000$




(b) $W = 250,000$




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




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

Figure 4.7.: Robustness of the approach regarding the number of workpieces ($S = 5$, bottleneck last)

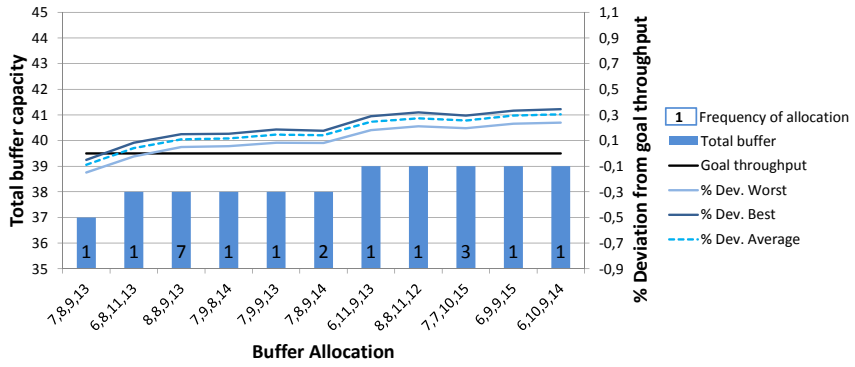



Figure 4.8.: Robustness of SRS ($S = 5$, $W = 250,000$, bottleneck last)

required to obtain robust results, because more different allocations are obtained (see Tables A.2 and A.3 in Appendix A respectively). Figure 4.7d 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 attain the goal throughput.

Figure 4.8 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 Section 4.2 ($W = 250,000$). Compared to the results in Figure 4.7b, 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, for SRS, a maximum deviation of 0.15% is observed. Consequently, this demonstrates that DS leads to more robust results than SRS.

### 4.4.2. 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 Section 4.3.2.

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 a better throughput unless the imbalance caused by the bottleneck is more than 20%. In the case of an imbalance of more than 20%, the capacity of the buffer, which is located farthest from the bottleneck, shall be decreased. The available buffer space shall be placed around the bottleneck. In-

Table 4.2.: Time saving potential of approximate solutions

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

feasible allocations are used as feasibility cuts (4.14), 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 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 Section 4.3.2 are of a different type as they provide (individual) lower bounds instead of only feasibility cuts.

Table 4.2 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 seconds 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 bounds generated from the subsystems as described in Section 4.3.2 instead of feasibility cuts or upper bounds from near-optimal solutions. For this reason, we omit further investigations of these upper bounds and feasibility cuts and focus on the lower bounds obtained from the subsystems.

### 4.4.3. 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 Section 4.4.1. 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, Section 4.4.1 shows that this sample size is not sufficient to obtain robust results. Therefore, further studies use samples with $W = 250,000$.
Table 4.3 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 seconds. Only two settings are solvable within this time limit using the original MIP or the Benders Decomposition approach with classical cuts.

Table 4.3.: Mean computation times (Exponential distribution)

| | | | Computation time (sec) | | | | |
|---|---|---|---|---|---|---|---|
| | | | | Benders Decomposition | | | |
| | | | Original | Cl. Cut | Comb. Cut | | |
| S | Bottleneck | W | formulation | | 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 |

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 and 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 4.3 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, Figure 4.9 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 a bottleneck at the end. To derive the lower bounds, we optimized four 2-station subsystems, three 3-station subsystems, and two 4-station subsys-

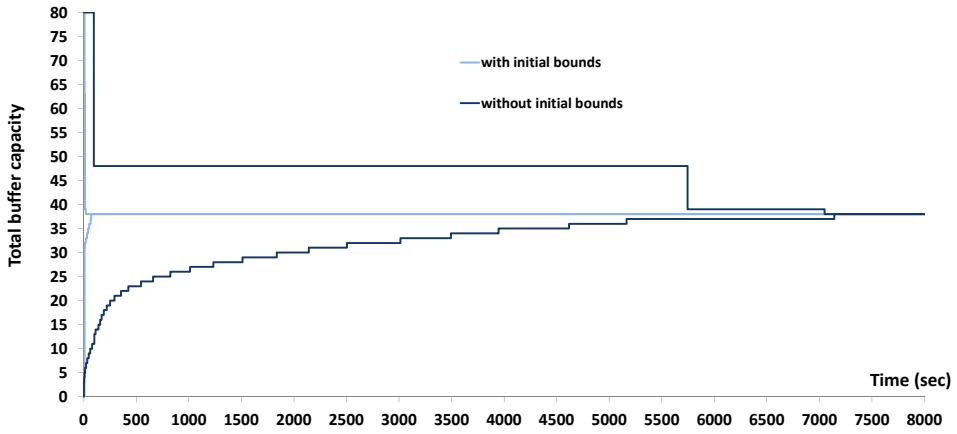



Figure 4.9.: Course of the lower and upper bounds during the solution process ($S = 5$, $W = 250,000$, bottleneck last)

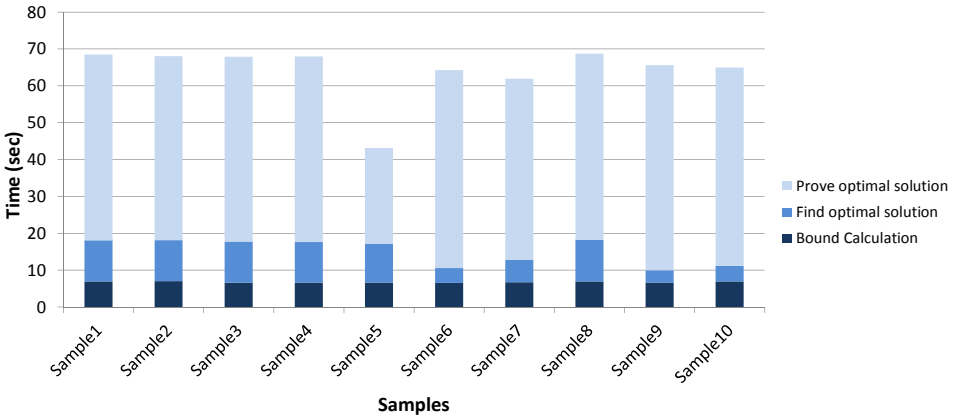



Figure 4.10.: Share of computation times for bound calculation and optimality proof ($S = 5$, $W = 250,000$, bottleneck last)

tems. The computation of the bounds is completed after 8 seconds, 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 seconds and is proven after 69 seconds. Without bounds, the upper bound drops in large steps until the optimal solution is found after 7051 seconds. This solution is proven to be optimal after 7141 seconds.

Figure 4.10 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 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.

### 4.4.4. 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 4.4. We generate 10 independent samples for each configuration.

Table 4.4.: 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 |

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 coefficients 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 4.5. 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 4.6, A.1, A.2, and A.3 in Appendix A. Tables 4.6, A.1, A.2, and A.3 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

Table 4.5.: Mean computation times (Erlang-k and Cox-2 distribution)

| Distribution | S | SCV | Bottleneck | Range of total buffer capacities | Computation time (sec) Benders Decomposition (Comb. Cut) Bounds | Total | Max. dev. from $TH^*$ (%) |
|---|---|---|---|---|---|---|---|
| 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 4.6.: Detailed results[1](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 |

are especially difficult to solve, the computation time takes more than 1 h on average.

Table 4.5 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 4.6, A.1, A.2, and A.3 show that it is reasonable to solve all subsystems, as even large subsystems improve the (aggregated) bounds on the buffer capacities.

The column "Max. dev. from $TH^*$" depicts the results of a throughput evaluation

[1]Tight bounds are marked in bold

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 is not attained and the smallest relative upward deviation if it is attained. The deviation for each buffer allocation is shown in Tables 4.6, A.1, A.2, and A.3. 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.

### 4.4.5. 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 an 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 4.4). We compare the results to allocations obtained from instances generated by a Generalized Erlang distribution based on identical parameters. This corresponds to the case where correlation is neglected and approximated by independent identically distributed processing times. The Generalized Erlang distribution may be interpreted as a random decision on the type for each processed workpiece at each station.

All instances are solved in less than 15 minutes. Further computational results are given in Table 4.7. Columns 2 to 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 substantial misallocation of buffer spaces. Therefore, it is

Table 4.7.: Detailed results (correlated processing times)

| | Correlation | | | Generalized Erlang | | |
|---|---|---|---|---|---|---|
| Sample | Total buffer capacity | Optimal allocation | Max. dev. from $TH^*$ (%) | Total buffer capacity | 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 |

important that correlations are considered in the solution approach, as it is possible with our approach.

### 4.4.6. 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 Figures 4.11 and 4.12).

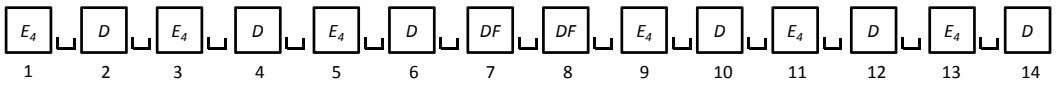



Figure 4.11.: Setting of the 14-station line

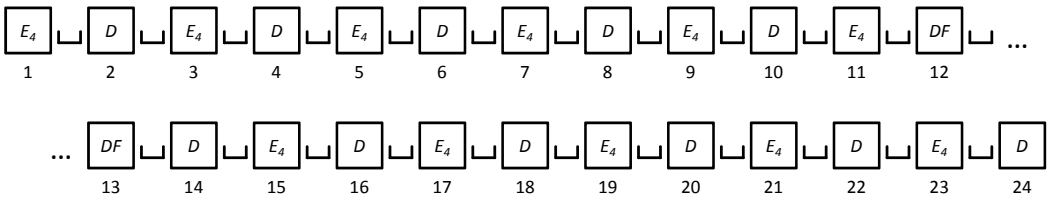



Figure 4.12.: Setting of the 24-station line

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 the 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 4.4).

Table 4.8.: Detailed results ($S = 14$)

| | Sample | Total buffer capacity | Optimal allocation | Computation times (sec) Subsystems | Total | Max. dev. from $TH^*$ (%) |
|---|---|---|---|---|---|---|
| **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 4.8 and 4.9 contain the results of this experiment. The algorithm solves instances with 14 stations within 310 seconds on average for TTF = 10 and TTR = 4. If TTF = 5 and TTR = 2, the algorithm takes 50 seconds on average. For the line with 24 stations, 14 hours 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 minutes 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

Table 4.9.: Detailed results ($S = 24$)

| | Sample | Total buffer capacity | Optimal allocation | Computation times (sec) Subsystems | Total | Max. dev. from $TH^*$ (%) |
|---|---|---|---|---|---|---|
| **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 | 0,0,0,0,0,0,1,0,0,1,3,7,0,3,2,0,0,0,0,0,0,0,0 | 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 |

bottleneck stations. At the beginning and at the end of the line, zero or only few buffer spaces are required. The last column 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 is not attained and the smallest relative upward deviation if it is attained. 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.

## 4.5. 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 compare 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.

# 5. Optimization of buffer allocations in flow lines with limited supply

*Co-authors:*

**Andrea Matta**
Shanghai Jiao Tong University, Department of Industrial Engineering and Management, Shanghai, P. R. China

**Raik Stolletz**
Chair of Production Management, Business School, University of Mannheim, Germany

*Working paper.*

*Abstract:*

The supply of flow lines is often assumed to be unlimited or to follow certain distributions. However, this assumption may not always be realistic because flow lines are usually an integral part of a supply chain where raw material is replenished according to some rule. We therefore include the limited supply into the optimization of buffer capacities in terms of an order policy.
To integrate this type of supply into an optimization model, we exploit the flexibility of a sample-based optimization approach. We develop an efficient rule-based local search algorithm that employs new individual lower bounds in order to determine the optimal buffer capacities of a flow line. Besides the efficiency of the proposed algorithm, the numerical study demonstrates that the order policy has a significant impact on the optimal buffer allocation.

## 5.1. Introduction

Flow lines consist of a number of stations that are arranged in series and separated by buffer spaces. Stochasticity in such lines can be caused by random machine breakdowns, uncertain times to repair, and random processing times. If buffer capacities are limited, blocking and starvation effects may occur. This may lead to a reduction of the throughput of the entire line. Allocating additional buffer capacities decouples the stations and therefore counteracts these effects. However, the average work-in-process in the line increases, which involves additional costs.

The decisions on the total quantity of buffer spaces and their allocation within the flow line, which balance the trade-off between resulting costs and obtained throughput, are known as the Buffer Allocation Problem (BAP). Multiple examples from the practice concerning this problem can be found in the literature. Most examples apply to the automobile industry (e.g. Li, 2013; Alden et al., 2006; Colledani et al., 2010), but lines from food industry (e.g. Liberopoulos and Tsarouhas, 2002), and other manufacturing applications (e.g. Burman et al., 1998) are also described. These articles demonstrate the potential of operations research methods in determining the optimal allocation of buffer capacities and report on the resulting benefits.

In the literature, the BAP is usually solved under the assumption of unlimited supply (Gershwin and Schor, 2000). To ensure unlimited supply in practice, large inventory levels in front of the first station are required to allow for stochastic effects in the line. Some articles take limited supply into account but assume that the arrival times of the workpieces are exogenously determined (e.g. Dallery and Gershwin, 1992; Matta, 2008) or that an additional station models the supply (e.g. Dallery and Gershwin, 1992; Helber et al., 2011). Yet this is not realistic, because independency of the system state and the arrival pattern is assumed. In reality, orders are placed depending on the inventory level in front of the first station.

Various problem formulations of the BAP with different objectives can be found in the literature. An overview on these objectives and the existing optimization approaches is given by Gershwin and Schor (2000) and Demir et al. (2014).

In the following, we review approaches that provide exact optimal solutions for the BAP or provide bounds on the buffer capacities. Exact analytical results are only available for very small lines under restrictive assumptions (see e.g. Buzacott, 1971). For longer lines, Matta (2008) proposes a mixed integer programming (MIP) formulation that uses sampling. Sampling approaches replace the stochastic elements by their sampled counterparts. They therefore allow for a large degree of flexibility. Hence, these approaches can be used for more realistic modeling of the

underlying problem. It is possible to allow for any distribution of processing times, times to failure, and repair times, as well as correlations therein. Moreover, the resulting performance measures are *sample-exact* and converge to the exact value provided that sample sizes are chosen sufficiently large. However, the corresponding sample-based MIP is only capable of solving very small instances with three stations. Alfieri and Matta (2012) introduce the concept of time buffers, which reduce the feasible region of the buffer capacities. Yet the derivation of the time buffers is only possible for small instances with three stations. Weiss and Stolletz (2015) consider a MIP formulation similar to Matta (2008). To accelerate the solution process, they propose a Benders Decomposition approach in combination with the generation of lower bounds derived from subsystems. They use the flexibility of the approach to demonstrate the impact of correlations on the optimal buffer allocation. The work of Shi and Gershwin (2014) is closely related because the proposed segmentation approach applies the concept of subsystems to estimate the buffer capacities.

Matta et al. (2014) describe a general methodology to derive simulation-optimization models in terms of mathematical programming. This concept is also applied for Base Stock Control Systems and Extended Kanban Control Systems (Pedrielli et al., 2015) and for the optimization of the number of pallets in ConWIP systems (Alfieri et al., 2015).

The main contribution of this paper is to incorporate limited supply in the form of an order policy into the optimization of the BAP in order to gain managerial insights into the allocation of buffer capacities under such realistic assumptions. Moreover, the impact of a policy-driven supply is demonstrated. Individual lower bounds on the buffer capacities and a rule-based local search algorithm are developed to efficiently and optimally solve the resulting problem. The rule-based local search algorithm first investigates promising solutions in the neighborhood of the current allocation under consideration of the individual lower bounds. To ensure optimality it then jumps to allocations in regions which have not been investigated before.

This paper is organized as follows. Section 5.2 introduces the assumptions of the flow line model and the decision problem. In Section 5.3, the individual lower bounds are presented. Section 5.4 describes the rule-based local search algorithm for buffer optimization. In Section 5.5, a numerical study on the performance of the search algorithm and the impact of limited supply on the flow line is provided. Finally, Section 5.6 presents the conclusion and suggestions for further research.

## 5.2. Model of the flow line

This paper considers the allocation of a minimum number of total buffer spaces while attaining a pre-defined goal throughput, which is known as the primal BAP (Gershwin and Schor, 2000).
Section 5.2.1 presents the decision problem and the underlying assumptions for the flow line model. The modeling and the assumptions with respect to the limited supply are explained in detail in Section 5.2.2.

### 5.2.1. Model assumptions and decision problem

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

- The flow line consists of $m = 1, \ldots, M$ stations in series, which process $W$ workpieces.
- The decision $X_m$ about the capacity of the buffer behind station $m$ is limited by $B_m$.
- An order policy is applied to manage the material supply to the first station, i.e., the supply is limited. Unlimited supply can be modeled by selecting adequate parameters for the policy.
- The buffer behind the last station is infinitely large, $B_M = \infty$. Thus this station cannot be blocked.
- The processing times of the workpieces at each station are generally distributed.
- The stations may be subject to operation-dependent failures. Times-to-failure and times-to-repair are generally distributed.
- 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 through the buffer are insignificant or are already included in the processing times.
- The performance of the line is measured with respect to the expected throughput E[TH($X_1, \ldots, X_{M-1}$)] and is evaluated under steady-state conditions.

Figure 5.1 shows an example of a flow line according to these assumptions. The

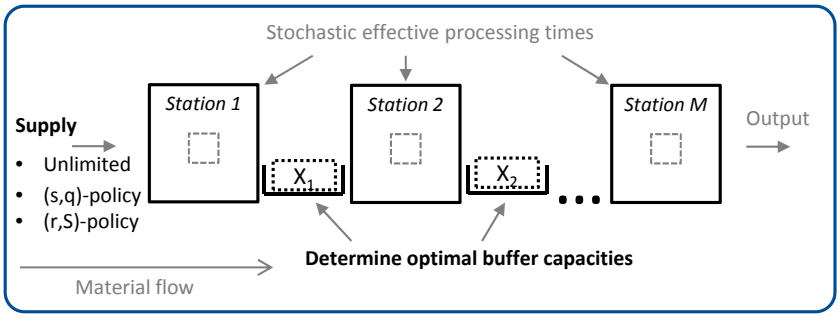



Figure 5.1.: Flow line under consideration

mathematical formulation of the decision problem is given in Formulation (5.1).

$$\min \sum_{m=1}^{M-1} X_m \tag{5.1a}$$

s.t.

$$E[TH(X_1,\ldots,X_{M-1})] \geq TH^* \tag{5.1b}$$

$$X_m \leq B_m, \qquad \forall m \tag{5.1c}$$

$$X_m \geq 0, \quad \text{integer}, \qquad \forall m \tag{5.1d}$$

The objective function (5.1a) minimizes the total buffer capacity in the line. Equation (5.1b) ensures that the goal throughput, $TH^*$, is attained. In Constraint (5.1c) the physical floor limitations are defined as upper bounds on the buffer capacities. Equation (5.1d) assures that the buffer capacity variables are non-negative and integer.

### 5.2.2. Supply of the first station

We assume that the supply of the first station is organized by an order policy which launches replenishment orders depending on the inventory position in front of the first station. The inventory position consists of the current inventory level and already placed orders, which have not been received yet (i.e., these orders did not arrive at the first station). Each order has a pre-defined lead time of $T$ time units. Such policies are e.g. described by Silver et al. (1998).

We test two types of inventory policies, the (s,q)-order policy and the (r,S)-order policy. However, any type of order policy can be used in our approach.

The *(s,q)-order policy* is based on a reorder point $s$, a constant order quantity $q$, and a lead time $T$. Whenever the inventory position of the storage in front of the

first station drops to the reorder point $s$ or below, an order of size $q$ is placed. Consequently, the inventory position must be continuously monitored. Each order requires a lead time of $T$ periods until delivery.
The *(r,S)-order policy*, in contrast, is based on periodic review and uses a review interval $r$, a lead time $T$, and an order-up-to-level $S$. An order is placed every $r$ periods. Each order requires a lead time of $T$ periods until delivery. The order quantity is chosen in such a way that the order raises the inventory position to the order-up-to-level $S$. In some cases, it is convenient to always order a multiple of a certain order quantity $q$ (e.g. truck loads) instead of ordering an arbitrary number of items. In this case, the order quantity is calculated by $\lceil \frac{S-\text{inventory position}}{q} \rceil \cdot q$. This formula will be used in what follows.
Whenever the inventory in front of the first station is empty, the starting time of the next workpiece is delayed at least until the next order arrives. Otherwise, processing at the first station begins when the previous workpiece leaves the station. Depending on the parameters of the (s,q) or the (r,S)-policy, there is a positive probability that the first station will starve, which is not possible when assuming unlimited supply.
This model is closely related to the models proposed in inventory literature. Axsäter and Rosling (1993) show that a flow line can be modeled as a series of installations which are supplied by order policies and model the interaction of two consecutive stations. The consideration of the limited supply of the line corresponds to an additional installation in front of the line. However, for these models the waiting and blocking times of the workpieces must be known to determine the lead time of the order policies. This is not the case under general assumptions because the waiting and blocking times depend on the buffer capacities and this relation cannot be expressed in a closed form. Therefore, modeling the flow line as a series of installations which are connected by order policies is not applicable. Consequently, approaches from the inventory literature cannot be applied.

## 5.3. Individual lower bounds on the buffer capacities

The BAP is an NP-hard problem (Smith and Cruz, 2005). The feasible region grows non-linear with the number of stations in the line. This complexity requires solution approaches that extensively reduce the size of the feasible region. Therefore, we develop new lower bounds on the optimal individual buffer capacities in order to reduce the solution space of the BAP.
Weiss and Stolletz (2015) develop *aggregate lower bounds* (ALB) for groups of

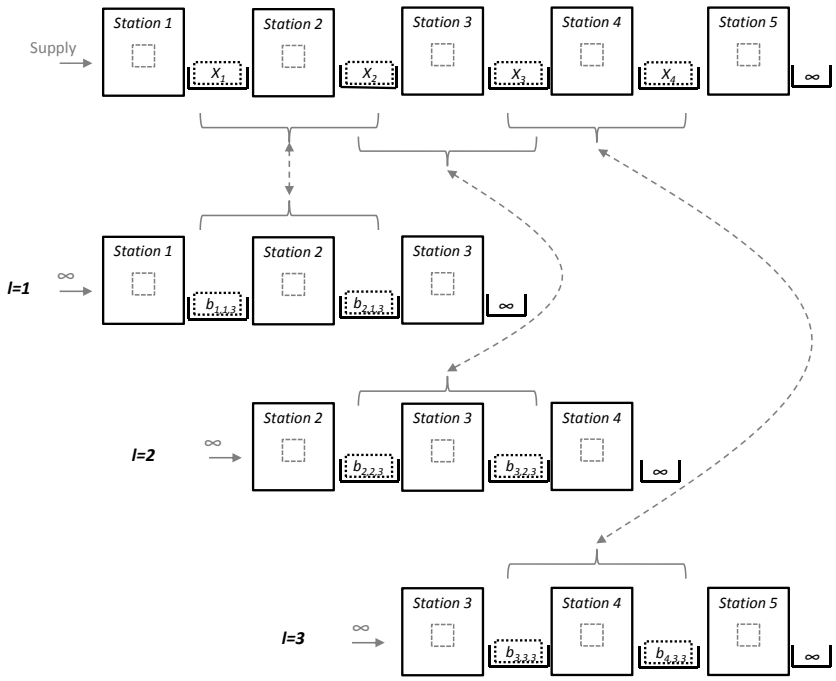



Figure 5.2.: All subsystems of size $i = 3$ for a line with $M = 5$ stations

buffers based on the optimization of subsystems of the line. However, using these aggregate bounds it is not clear how many buffer spaces have to be assigned to which individual buffer. *Individual lower bounds* (ILB) restrict the solution space of the BAP more extensively than ALBs. The idea of generating individual lower bounds consists of three steps and is outlined in what follows. First, ALBs are derived for each subsystem. The capacities are reallocated based on the ALB within the subsystem in a second step. In the third step an ILB is determined as the minimum over all subsystems.

**Generation of aggregate lower bounds**

Figure 5.2 depicts the decomposition of the flow line into subsystems as introduced in Weiss and Stolletz (2015). The isolated optimization of a subsystem results in lower bounds that are valid for *groups* of buffers in the original line, but do not hold for individual buffers. We therefore refer to these bounds as *aggregate lower bounds* (Weiss and Stolletz, 2015).

Each subsystem consists of $i$ stations and is assumed to operate independently of the remaining stations of the line. Blocking and starvation that may occur in the original line because of the interaction of stations not included in the subsystem or the limited supply in front of the first station are neglected. Consequently, the isolated optimization of a subsystem results in the same or less total buffer capacity than in the entire line (Weiss and Stolletz, 2015). The optimal buffer capacity

Table 5.1.: Notation for the calculation of lower bounds

| | |
|---|---|
| **Indices** | |
| $m = 1, \ldots, M$ | Stations in the flow line |
| $i = 2, \ldots, M-1$ | Sizes of the subsystems |
| $l = 1, \ldots, M-i+1$ | Subsystems of size $i$ |
| **Parameters** | |
| $TH^*$ | Goal throughput |
| $B_m$ | Maximum capacity of the buffer behind station $m$ |
| **Real-valued decision variables** | |
| $E[TH(X_1, \ldots, X_{M-1})]$ | Expected throughput obtained with buffer allocation $X_1, \ldots, X_{M-1}$ |
| **Integer decision variables** | |
| $X_m$ | Buffer capacity behind station $m$ |
| $b_{m,l,i}$ | Buffer capacity behind station $m$ in subsystem $l$ of size $i$ |
| $b^{m'}_{m,l,i}$ | Buffer capacity behind station $m$ in the allocation that contains the individual lower bound for buffer $m'$ |
| $b^*_{m,l,i}$ | Individual lower bound for the buffer behind station $m$, originating from subsystem $l$ of size $i$ |

of station $m$ in the isolated subsystem $l = 1, \ldots, M-i+1$ of size $i$ is denoted by $b_{m,l,i}$. The allocated total buffer capacity in the subsystem, $\sum_{m=l}^{l+i-2} b_{m,l,i}$, is a lower bound for the capacities of the respective buffers in the original line, $\sum_{m=l}^{l+i-2} X_m$ (Weiss and Stolletz, 2015), see Table 5.1 for the used notation.

**Reallocation of buffer capacities within a subsystem**

We calculate the ILB for a buffer $m'$ in subsystem $l$ of size $i$ using the ALB $\sum_{m=l}^{l+i-2} b_{m,l,i}$. The idea is to reallocate the buffer capacities of the ALB such that the capacity $b^{m'}_{m',l,i}$ for buffer $m'$ is minimized under consideration of the throughput constraint (see the mathematical program (5.2)).

$$b^*_{m',l,i} = \min\ b^{m'}_{m',l,i} \tag{5.2a}$$

$$\sum_{m=l}^{l+i-2} b^{m'}_{m,l,i} \geq \sum_{m=l}^{l+i-2} b_{m,l,i} \tag{5.2b}$$

$$E[TH(b^{m'}_{l,l,i}, \ldots, b^{m'}_{l+i-2,l,i})] \geq TH^* \tag{5.2c}$$

$$b^{m'}_{m,l,i} \leq B_m, \qquad \forall m = l, \ldots, l+i-2 \tag{5.2d}$$

$$b^{m'}_{m,l,i} \geq 0, \quad \text{integer}, \qquad \forall m \tag{5.2e}$$

The objective function (5.2a) minimizes the capacity of buffer $m'$ in the subsystem $l$ to obtain an ILB for $m'$. In Constraint (5.2b) it is ensured that the total number of buffer spaces of the candidate allocation is larger or equal to the ALB derived

from the optimization of the subsystem $l$. Additionally, the goal throughput has to be attained by the expected throughput of the subsystem obtained with allocation $b^{m'}_{l,l,i}, \ldots, b^{m'}_{l+i-2,l,i}$. This is assured by Constraint (5.2c). Constraints (5.2d) ensure that the previously defined maximum buffer capacity, $B_m$, is complied with. In Constraints (5.2e), it is specified that the buffer capacities are non-negative and integer. Note that the sum of the ILBs, $\sum_{m=l}^{l+i-2} b^*_{m,l,i}$, from subsystem $l$ of size $i$ is in general smaller than the corresponding ALB.

**Theorem 5.1.** *$b^*_{m',l,i}$ is an individual lower bound for the capacity of buffer $m'$.*

*Proof.* Weiss and Stolletz (2015) prove that $\sum_{m=l}^{l+i-2} b_{m,l,i}$ is an ALB for the total capacity of buffers $l, \ldots, l+i-2$ with $l = 1, \ldots, M-i+1$ and $i = 1, \ldots, M-1$. Constraint (5.2b) holds because $\sum_{m=l}^{l+i-2} b^{m'}_{m,l,i} < \sum_{m=l}^{l+i-2} b_{m,l,i}$ violates the ALBs. Constraints (5.2c)-(5.2e) formulate Equations (5.1b)-(5.1d) for the subsystems and therefore only exclude candidate allocations that are also excluded when calculating the ALBs using Formulation (5.1). Consequently, the feasible region of the mathematical program (5.2) consists of all optimal allocations for subsystem $l$ of size $i$. The objective function (5.2a) minimizes the capacity of buffer $m'$. Thus, when solving the mathematical program (5.2), the result will be a feasible buffer allocation for the subsystem with minimum capacity of buffer $m'$, i.e., a lower bound for the capacity of buffer $m'$. □

**Derivation of minima**

To obtain all ILBs from a subsystem $l$ of size $i$, the mathematical program must be solved for each buffer $m' = l, \ldots, l+i-2$. Moreover, we calculate the ILBs resulting from different subsystems $l = 1, \ldots, i-1$ of sizes $i = 3, \ldots, M-1$, starting with $m' = 1, l = 1$, and $i = 3$. Consequently, several ILBs are obtained for each buffer $m'$.

Because the different ILBs for a buffer $m'$ dominate each other, only the most restrictive ILB, i.e. the maximum value, is used for the optimization of the entire line. The constraints resulting from the ILBs,

$$\max_{l,i} b^*_{m,l,i} \leq X_m \quad \forall m, \tag{5.3}$$

can also be used iteratively in the calculations of the ALBs and ILBs for all buffers of the subsystems $l+1, l+2, \ldots$ of size $i$ as well as for the calculations of larger subsystems $i+1, \ldots, M-1$.

In general, these bounds can be calculated with any buffer allocation algorithm because they can be derived by simply optimizing different subsystems of the line.

Additionally, such bounds can speed up different heuristic and exact solution approaches.

## 5.4. Rule-based local search algorithm

The rule-based local search (RBLS) algorithm solves the BAP under consideration of the pre-calculated lower bounds on the individual buffer capacities. Thereby, the RBLS algorithm iteratively applies a generative (see Section 5.4.1) and a sample-based evaluative method (see Section 5.4.2) as depicted in Figure 5.3 to determine the optimal solution. The exchange of information on feasibility and optimality between generative and evaluative method is ensured by feasibility cuts and upper bounds. This algorithm yields *sample-exact* buffer capacities, which converge to the exact optimum for sufficiently large samples.

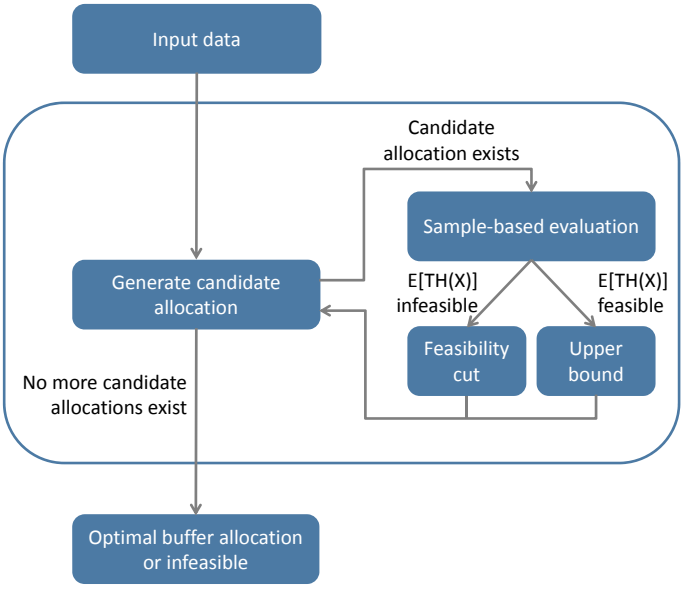



Figure 5.3.: Overview of the RBLS algorithm

### 5.4.1. Generation of candidate allocations

Generating a new buffer candidate allocation is a non-trivial task because of the complex relation between buffer spaces and throughput, which cannot be expressed in a closed form under general assumptions. We therefore develop a rule-based local search for the generation of candidate allocations.

The maximum buffer capacities, $B_1, \ldots, B_{M-1}$, are defined by the user and serve as a start solution. The generative method systematically generates new candidate

allocations under consideration of the pre-calculated lower bounds and the results of the evaluation. Three cases can be distinguished.

(I) If the evaluation of the current candidate allocation results in a feasible throughput, a new candidate allocation is generated by reducing the capacity of one of the buffers by one. To decide which buffer capacity is to be reduced, a buffer selection criterion uses information from the evaluation. Several criteria have been tested (see Section 5.5.1). The capacity of the buffer selected by the chosen selection criterion is reduced by one.

(II) If the evaluated allocation is infeasible, the evaluation data of the last feasible allocation is used to determine another buffer, if available, according to the selection criterion. The respective buffer capacity of the last feasible allocation is reduced by one.

(III) If the evaluated allocation is infeasible and all its neighborhood candidates already have been evaluated, an artificial allocation with a total buffer capacity of the current upper bound-1 is selected. We choose the allocation in the middle of our candidate allocation vector.

Cases (I) and (II) represent the local search of the algorithm. Case (III) represents a search in the global region to ensure that the algorithm finds the optimal solution and is not trapped in local optima.
Whenever all candidates have either been evaluated or excluded by bounds or cuts, i.e., no further candidate allocations exist, the last upper bound is equal to the optimal buffer allocation. If no upper bound was detected during the solution process, the problem is infeasible.
Despite the rule-based local search, any other algorithm for the generation of candidate allocations can be used, if it ensures to generate not only allocations in the neighborhood of current candidate allocations but also in the entire feasible region.

### 5.4.2. Sample-based evaluation and exchange of information

The candidate allocations are evaluated by a sampling algorithm with respect to the throughput which is adapted from Chen and Chen (1993). Sample-based approaches model the flow of a large number of workpieces throughout the line. The random processing times, times to failure, and repair times are replaced by sampled effective processing times, which are generated by Descriptive Sampling (Saliby, 1990a). See Weiss and Stolletz (2015) for a detailed description of the sampling algorithm for the case of unlimited supply. We extend this algorithm to consider (r,S)

and (s,q)-order policies modeling the supply of the first station (see Appendix B).
If the throughput, $E[TH(X_1, \ldots, X_{M-1})]$, resulting from the evaluation is lower than the goal throughput, $TH^*$, the evaluated candidate allocation is infeasible. This candidate allocation as well as all dominated allocations are then excluded by *feasibility cuts* which are added to the rule-based local search. An allocation is dominated if all its buffer capacities are smaller or equal to the respective buffer capacities in the candidate allocation, see Weiss and Stolletz (2015). The lower bound on the total buffer capacity is (implicitly) increased if all candidates of a certain total number of buffer spaces are infeasible, i.e., all corresponding feasibility cuts have been generated.
If the candidate allocation is feasible, the *upper bound* on the total buffer capacity is updated to exclude all allocations with a higher or equal total number of buffer spaces.

## 5.5. Numerical Study

The algorithms are implemented in C++. Gurobi 5.0, with default settings and callbacks, is used to solve the mathematical programs described in Section 5.3. Callbacks are used to invoke the evaluation routine whenever Gurobi finds an incumbent solution. If the evaluation routine returns an infeasible throughput, the incumbent is rejected. The numerical study is performed on an Intel Core i7-3930K with 6x 3.2 GHz and 32 GB RAM.
In all instance types, the total number of workpieces of a sample is set to $W = 250,000$ and the warm-up phase is selected as $W_0 = 2,000$. We generate 10 independent samples for each configuration. The detailed description of the test instances is given in the respective sections.
We first compare different selection criteria as part of the RBLS algorithm in Section 5.5.1. Section 5.5.2 investigates the performance of the RBLS algorithm and the ILBs. In Section 5.5.3, the impact of the order policies on the optimal buffer allocation is evaluated.

### 5.5.1. Impact of different buffer selection criteria

Table 5.2 shows ten different selection criteria that were implemented within the RBLS algorithm. Both, criteria from literature and new criteria are tested.

Table 5.2.: Average computation times with different selection criteria (10 samples)

| | Bottleneck last | | Bottleneck middle | |
|---|---|---|---|---|
| **Criterion** | Av. comp. time (s) | Dev. from best (%) | Av. comp. time (s) | Dev. from best (%) |
| **Vergara and Kim (2009)** | | | | |
| Number of blocking events | 2616 | 1 | 1496 | 3 |
| Blocking time | **2581** | – | **1458** | – |
| Number of starvation events | 2696 | 5 | 1519 | 4 |
| Starvation time | 2597 | 1 | **1459** | – |
| Number of blocking and starvation events | 2621 | 2 | 1501 | 3 |
| Blocking and starvation time | 2727 | 6 | 1531 | 5 |
| **New** | | | | |
| Net blocking time | 2909 | 6 | 1531 | 5 |
| Net starvation time | 2644 | 2 | 1486 | 2 |
| **Li and Meerkov (2009)** | | | | |
| Equal protection criterion | 3375 | 31 | 1933 | 33 |
| Buffer half-full criterion | 2899 | 12 | 1696 | 16 |

**Criteria proposed by Vergara and Kim (2009)**

Vergara and Kim (2009) propose several criteria based on blocking and starvation. The *number of blocking events* at a station $m$ and the *blocking time* of a station $m$ respectively, are related to the buffer behind station $m$. In contrast, starvation is caused by the buffer in front of station $m$. Therefore, the *number of starvation events* and the *starvation time*, respectively, are related to the upstream buffer, i.e. the buffer behind station $m-1$. The *number of blocking and starvation events* and the *blocking and starvation time* with respect to buffer $m$ respectively, are a combination of the above blocking and starvation criteria. In all cases, the capacity of the buffer with the lowest value of the criterion is decreased by one.

**New criteria**

Additionally, we test two criteria, which are also based on blocking and starvation times but have not been reported in literature yet. The *net blocking time* criterion associated with the buffer behind station $m$ only considers blocking times that are caused by this buffer. This means that blocking times are only considered if station $m+1$ is not blocked at the same time. The *net starvation time* criterion analogously considers only starvation times of station $m$ if station $m-1$ is not starved at the same time.

**Criteria proposed by Li and Meerkov (2009)**

The *equal protection criterion* described in Li and Meerkov (2009) is based on the observation that buffer allocations with equal protection of station $m$ against blocking and starvation are good candidates for the optimal allocation. As a consequence, Li and Meerkov (2009) propose to calculate the indicator $E[WIP_m] - (X_{m+1} - E[WIP_{m+1}])\ \forall m = 1, \ldots, M-2$ where $E[WIP_m]$ is the expected work-in-process (WIP) in the buffer behind station $m$. This measures the balance of the expected WIP before station $m+1$, $E[WIP_m]$, and the expected number of free buffer spaces behind station $m+1$, $X_{m+1} - E[WIP_{m+1}]$. The idea is to enable a smooth flow by providing sufficient space behind the station for the expected amount of material in front of the station. Li and Meerkov (2009) describe the application of this criterion for the case of maximizing the throughput, subject to a constant total buffer capacity. We adapt the procedure for the problem of capacity minimization subject to a throughput constraint as follows. Let $m'$ be the station with the largest absolute value of the indicator. If the value of the indicator is positive, the capacity of the buffer behind station $m'-1$ is decreased by one, because the expected WIP in front of the station is too high compared to the expected number of free buffer spaces. Otherwise, the capacity of the buffer behind station $m'$ is decreased by one, because the expected number of free buffer spaces is too high compared to the expected WIP in front of the station.

The *buffer half-full criterion* arises from the observation that a full buffer protects best against starvation of the succeeding station, while an empty buffer protects best against blocking of the preceding station (Li and Meerkov, 2009). Thus, a buffer which is on average half-full is a compromise between the two extreme cases.

To compare these criteria, we use a line with $M = 7$ stations, unlimited supply, and exponentially distributed processing times with a base processing rate of 7 units per time. The bottleneck is located either at the station in the middle of the line or at the last station of the line and has a processing rate of 6 units per time. The capacity of each buffer is limited to $B_m = 20$. The goal throughput is set to 5.776.

Table 5.2 shows the average computation times (resulting from 10 different samples) of the RBLS algorithm in combination with the different criteria (2nd and 4th column). Note that the given computation times are generated with the RBLS algorithm in combination with ILBs. We have chosen the lowest computation times (bold) as the reference values for the calculations of the deviations in columns 3 and 5 for each type of instance. The lowest computation times for both bottleneck locations are obtained by the blocking time criterion. For a bottleneck in the middle of

the line, the starvation time criterion results in the same average computation times. All other criteria containing blocking or starvation times also result in low computation times with only 1-6% deviation compared to the blocking time criterion. Consequently, the RBLS algorithm can be combined with any of these criteria. In contrast, the buffer half-full criterion and the equal protection criterion result in a rather poor performance. In the following experiments, we apply the blocking time criterion of Vergara and Kim (2009). However, the superiority of certain selection criteria may depend on the structure of the instance chosen.

### 5.5.2. Impact of individual bounds and the rule-based local search algorithm

We first analyze the RBLS algorithm (based on the blocking time criterion) for unlimited supply. This allows for a comparison with the results of the Benders Decomposition in Weiss and Stolletz (2015). Both optimization algorithms are executed with ALBs (proposed by Weiss and Stolletz, 2015) and the new ILBs (as developed in Section 5.3). The experiments are based on the numerical study of Weiss and Stolletz (2015). All instances investigate flow lines with $M = 7$ stations. The capacity of each buffer is limited to $B_m = 20$. The bottleneck is located either at the station in the middle of the line or at the last station of the line. We test instances with Erlang-k, Cox-2, and exponentially distributed processing times. Table 5.3 shows the parameters which change for the different distributions.

Table 5.3.: Parameter settings of the test cases

| Processing time distribution | Erlang-k | Cox-2 | Exponential |
|---|---|---|---|
| Base processing rate | 0.5 | 0.5 | 7.0 |
| Processing rate of bottleneck | 0.45 | 0.45 | 6.0 |
| Goal throughput $TH^*$ | 0.405 | 0.405 | 5.776 |
| Squared coefficient of variation (SCV) | 0.25; 0.5 | 1.0; 2.0 | 1.0 |

Table 5.4 presents the average computation times resulting from 10 different samples for each of the different types of instances. The first three columns define the instance type. Column four contains the average computation time of the Benders Decomposition with ALBs as proposed in Weiss and Stolletz (2015). The fifth column depicts the results of the Benders Decomposition extended by the ILBs. The sixth and the seventh column consider the average computation times of the RBLS

Table 5.4.: Performance comparison of the solution methods (average of 10 samples per test case)

| | | | Average computation time (min) | | | | |
|---|---|---|---|---|---|---|---|
| | | Bottle- | Benders Decomposition | | RBLS algorithm | | |
| Distribution | SCV | neck | ALB | ILB | ALB | ILB | % eval. |
| Exponential | 1.0 | middle | 100 | 62 | 45 | 24 | 0.67 |
| Exponential | 1.0 | last | 125 | 76 | 81 | 43 | 1.11 |
| Erlang-4 | 0.25 | middle | $< 5$ | $< 5$ | $< 5$ | $< 5$ | 0.03 |
| Erlang-4 | 0.25 | last | $< 5$ | $< 5$ | $< 5$ | $< 5$ | 0.03 |
| Erlang-2 | 0.5 | middle | $< 5$ | $< 5$ | $< 5$ | $< 5$ | 0.03 |
| Erlang-2 | 0.5 | last | $< 5$ | $< 5$ | 6 | $< 5$ | 0.04 |
| Cox-2 | 1.0 | middle | 30 | 17 | 16 | 10 | 0.23 |
| Cox-2 | 1.0 | last | 40 | 22 | 20 | 11 | 0.26 |
| Cox-2 | 2.0 | middle | 101 | 74 | 29 | 23 | 0.65 |
| Cox-2 | 2.0 | last | 775 | 375 | 104 | 76 | 2.02 |

algorithm with ALBs and with ILBs respectively. The eigth column shows the proportion of evaluated allocations of the RBLS algorithm with ILBs from a total of $21^6 = 4084101$ possible allocations.

The results show that ILBs significantly improve the computation times of both the Benders Decomposition and the RBLS algorithm. Both algorithms solve the instances with Erlang-k distribution within a few minutes. For instances with exponentially or Cox-2-distributed processing times, a reduction of more than 65% of the computation time is achieved. The most difficult types of instances are those with Cox-2 distributed processing times, an SCV of 2.0, and a bottleneck at the end of the line. The Benders Decomposition with ALBs takes on average 775 minutes to solve one instance of this type. With ILBs this time is reduced to 375 minutes. The RBLS algorithm with ALBs takes on average 104 minutes. This can be reduced to 76 minutes when applying ILBs.

The comparison of the computation times of the Benders Decomposition and the RBLS algorithm, independently of the considered lower bounds, reveal that the RBLS algorithm improves the computation times of the difficult instances. In particular, the instances with Cox-2 distributed processing times with an SCV of 2.0 and a bottleneck at the end are solved within 375 minutes with a Benders Decomposition (with ILBs) and this is reduced to 76 minutes by the RBLS algorithm. The computation times of instances with Erlang-k-distributed processing times remain roughly the same. The required computation time is only of the order of a few minutes and therefore acceptable for both approaches. The number of evaluations

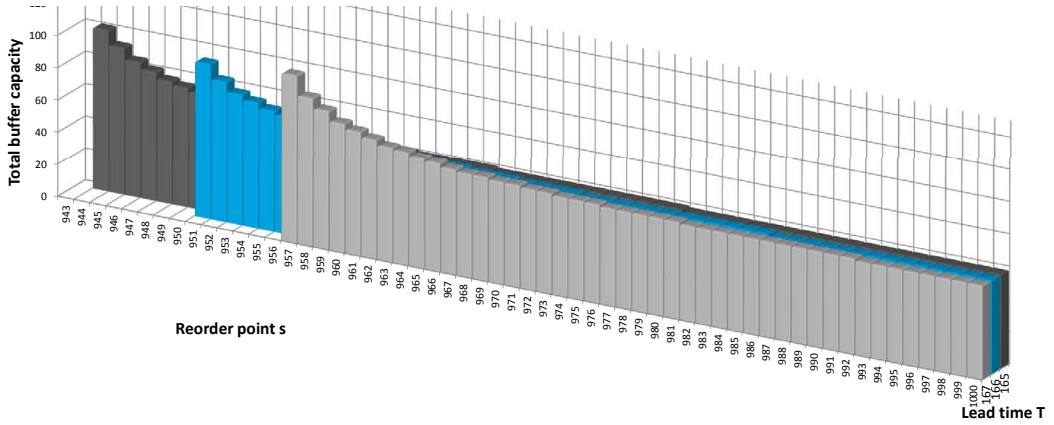



Figure 5.4.: Required total buffer capacity depending on the reorder point and the lead time for $q = 200$

during the execution of the RBLS algorithm with ILBs is very small with a maximum of 2% of the candidate allocations in the case of Cox-2-distributed processing times, an SCV of 2.0, and a bottleneck at the end of the line.

### 5.5.3. Impact of supply patterns

To investigate the impact of the order policies, we first optimize the total buffer capacity for a line supplied by a given (s,q)-order policy with varying reorder points $s$ and lead times $T$. The order quantity is set to $q = 200$ because preliminary studies revealed that the order quantity only has a minor influence on the optimal buffer allocation. Secondly, the optimal total buffer capacity for a line supplied by a given (r,S)-order policy with varying review intervals $r$, order-up-to levels $S$, and lead times $T$ is investigated. Finally, we present a study on the impact of the order policies on the computational performance.

In this section, we use an instance with exponentially distributed processing times and a bottleneck at the end of the line. This line has been described in the previous experiment. The maximum capacity, $B_m$, is set to the number of workpieces, $B_m = 250,000$. This corresponds to infinite buffers, i.e., we do not restrict the buffer capacities.

Figure 5.4 shows the optimal total buffer capacities for (s,q)-policies with $s = 943, \ldots, 1000$ and $T = 165, \ldots, 167$. Thus a total of 174 test cases is optimized. It can be observed that the required total buffer capacity increases exponentially with increasing $T$ and decreasing $s$ respectively. Moreover, there are certain values of $s$ and $T$ respectively for which the total buffer capacity cannot compensate the

lack of material induced by the pre-defined parameters of the order policy. Hence, there exist no feasible buffer allocations with respect to the goal throughput. The figure includes no bars for these cases. Out of the 174 test cases, 23 cases are infeasible. The larger the lead time $T$, the larger must be the reorder point $s$ selected in order to attain the goal throughput. If $s$ is chosen large or $T$ is chosen small enough, this corresponds to an unlimited supply, i.e., the optimal buffer capacity for the case of limited supply converges to the optimal solution with unlimited supply for increasing $s$ and decreasing $T$ respectively.

Table 5.5.: Optimal buffer allocations for selected (s,q)-policies with $q = 200$

| s | T | $\sum X_m$ | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | s | T | $\sum X_m$ | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1000 | 165 | 58 | 8 | 8 | 9 | 9 | 10 | 14 | 944 | 165 | 100 | 34 | 10 | 9 | 9 | 14 | 24 |
| 1000 | 166 | 58 | 8 | 8 | 9 | 9 | 10 | 14 | 951 | 166 | 96 | 25 | 16 | 9 | 9 | 13 | 24 |
| 1000 | 167 | 58 | 8 | 8 | 9 | 9 | 10 | 14 | 957 | 167 | 104 | 39 | 10 | 7 | 10 | 12 | 26 |

Table 5.5 depicts the resulting buffer allocations for selected (s,q)-order policies. The first group (columns 1 to 9) includes parameter choices that reflect unlimited supply. The allocations for unlimited supply ($s = 1000$) remain the same for varying lead time $T$. The second group (columns 10 to 18) corresponds to the lowest reorder points with a feasible solution for different lead times $T$. It can be seen that most of the capacity is located in front of the line and at the bottleneck, i.e., at the end of the line. The reason for adding buffer capacities in front of the line is that the lack of material induced by the limited supply is compensated by the additional buffer capacities. These capacities allow workpieces to already enter the line which subsequently triggers earlier replenishment.

Figures 5.5 and 5.6 show the optimal total buffer capacities for (r,S)-policies with $T = 165, \ldots, 167$ and $S = 1143, \ldots, 1200$ for $r = 35$ and $r = 40$ respectively. Thus, the buffer allocations for 348 test cases are optimized. The order quantity is set to $q = 1$, i.e. less than truck loads are allowed. Similarly to the (s,q)-policy, the total buffer capacity increases with increasing $r$, increasing $T$, and decreasing $S$. This quickly leads to infeasibility. Out of the 348 test cases, 78 cases are infeasible. Moreover, if $r$ and $T$ are chosen small enough and $S$ is chosen large enough, unlimited supply is obtained.

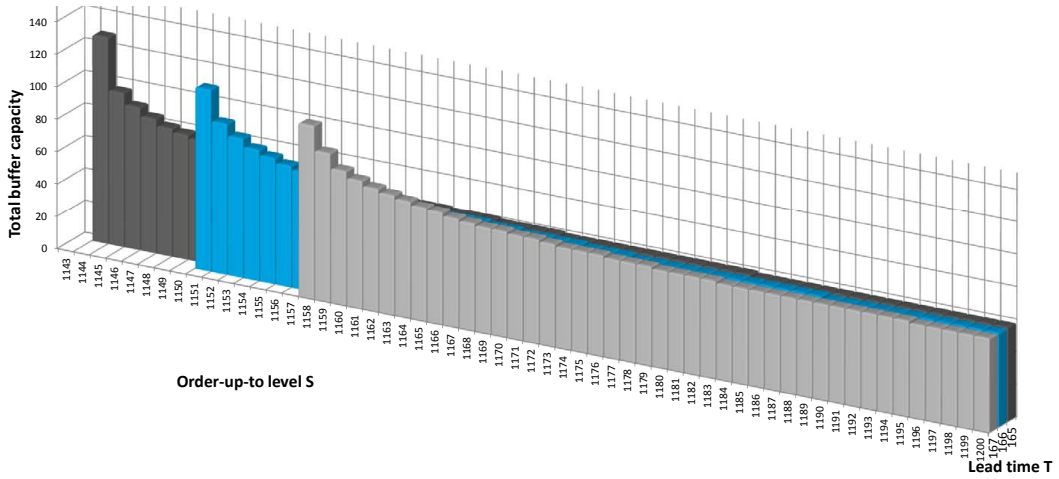



Figure 5.5.: Required total buffer capacity depending on the lead time and the order-up-to level for $r = 35$

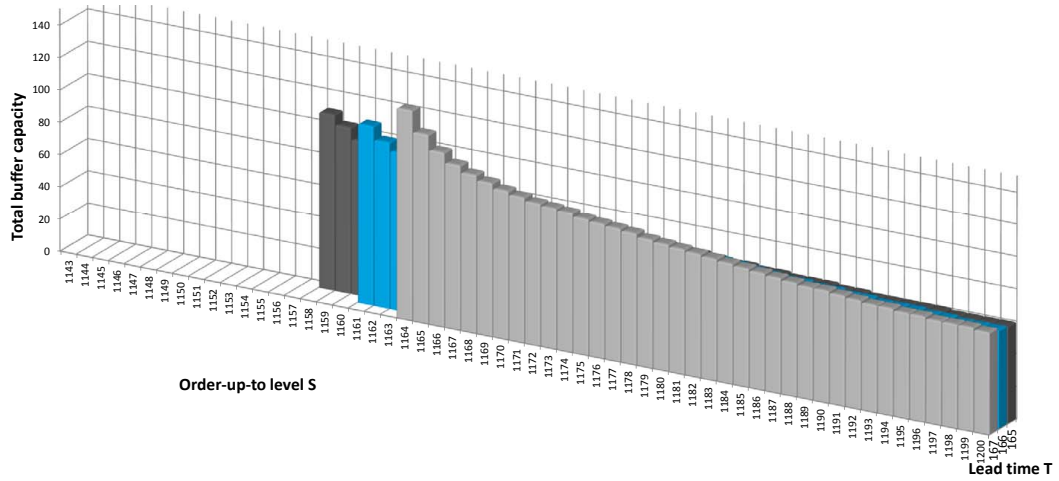



Figure 5.6.: Required total buffer capacity depending on the lead time and the order-up-to level for $r = 40$

Table 5.6.: Optimal buffer allocations for selected (r,S)-policies with $r = 35$

| S | T | $\sum X_m$ | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | S | T | $\sum X_m$ | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1200 | 165 | 58 | 8 | 8 | 9 | 9 | 10 | 14 | 1144 | 165 | 127 | 49 | 7 | 11 | 13 | 8 | 39 |
| 1200 | 166 | 58 | 8 | 8 | 9 | 9 | 10 | 14 | 1151 | 166 | 112 | 35 | 11 | 13 | 15 | 17 | 21 |
| 1200 | 167 | 58 | 7 | 9 | 9 | 9 | 10 | 14 | 1158 | 167 | 107 | 29 | 21 | 10 | 8 | 11 | 28 |

Table 5.6 presents the resulting buffer allocations for selected (r,S)-order policies with $r = 35$. The first group (columns 1 to 9) includes parameter choices that reflect unlimited supply. The allocations for unlimited supply ($S = 1200$) remain almost the same for varying lead times $T$. The second group (columns 10 to 18) corresponds to the lowest order-up-to-level $S$ with a feasible solution for different lead times $T$. It can be observed that the structure of the optimal allocations is similar to the case of the (s,q)-order policies.

Table 5.7.: Impact of neglecting limited supply

| s | T | Dev. from goal throughput (%) |
|---|---|---|
| 944 | 165 | -1.08 |
| 951 | 166 | -1.03 |
| 957 | 167 | -1.04 |

(a) (s,q)-order policies

| S | T | Dev. from goal throughput (%) |
|---|---|---|
| 1144 | 165 | -1.16 |
| 1151 | 166 | -1.08 |
| 1158 | 167 | -1.04 |

(b) (r,S)-order policies ($r = 35$)

The impact of neglecting the limited supply is illustrated in Table 5.7. The optimal allocations for the cases of unlimited supply from Tables 5.5 and 5.6 are evaluated under consideration of the order policies given in columns 1 and 2. Column 3 shows the deviation (in %) of the resulting throughput from the goal throughput. It can be seen that this deviation is larger than 1% in all cases. Hence, neglecting limited supply may result in buffer allocations that do not fulfill the pre-defined throughput goals. These results clearly demonstrate the need for solution approaches considering the supply patterns in front of the first station.

Figure 5.7 displays the impact of the order policies on the computation time. The computation times are obtained from an optimization run with maximum capacity $B_m = 20$. Each point in the figure corresponds to a buffer optimization with given policy parameters. These policies are combined into groups with respect to the lead

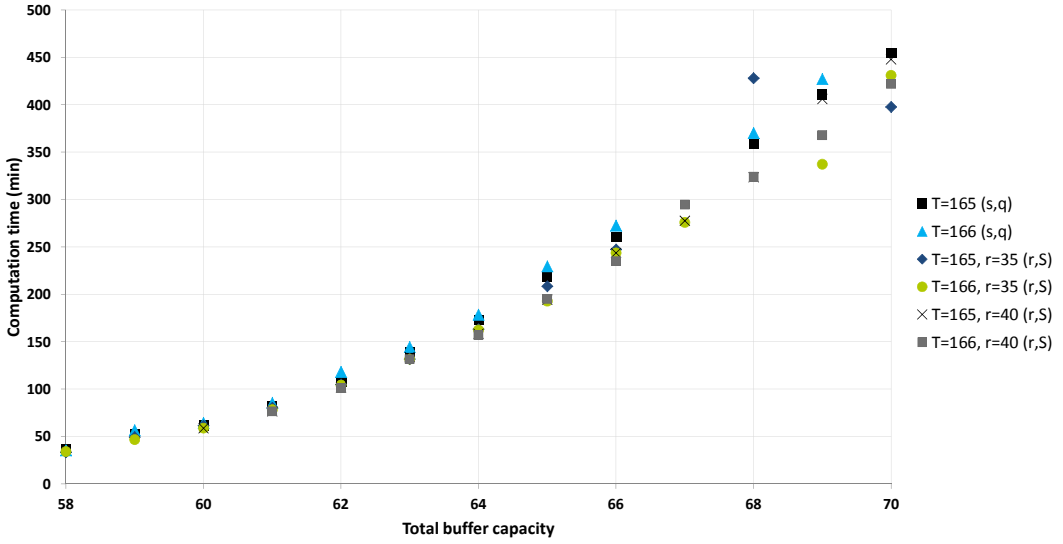



Figure 5.7.: Computation times in relation to the total buffer capacity of the optimal allocation

time $T$. The reorder point $s$ and the order-up-to level $S$, respectively, determine the total buffer capacity on the x-axis, i.e., they are only implicitly considered in the figure. With changing parameters, the total buffer capacity increases or decreases as pointed out in the previous experiments. Therefore, Figure 5.7 shows the computation times in relation to different optimal total buffer capacities. The different curves of computation times induced by the (s,q)-order policies and the (r,S)-order policies do not differ significantly, which supports the observation that the policy parameters itself have low impact on the computation time. Moreover, it can be observed that the computation time increases with the optimal total buffer capacity.

## 5.6. Conclusion and further research

In this paper, we develop individual lower bounds for the buffer capacities in flow lines. These bounds are derived by dividing the original system into subsystems and exploiting the fact that the subsystems are easier to solve. They can be applied in combination with any optimization algorithm for the BAP.
Furthermore, we develop a rule-based local search algorithm that uses the bounds to optimally and efficiently solve the BAP with limited supply. This algorithm iteratively decreases the total buffer capacity based on the results of throughput evaluations. We compare several types of criteria to select the buffer whose capacity is decreased.

Our numerical study shows that the application of both the individual bounds and the rule-based local search algorithm leads to substantial reductions in computation time. In addition, the numerical study reveals a significant impact of the limited supply on the optimal buffer capacity. Depending on the policy parameters, the optimal total buffer capacity increases exponentially. Thus, unless supplying the line with infinite material is not expensive, this work shows that the BAP cannot neglect the order policy governing the release of parts into the system.
Further research should extend this solution approach to take into account more complex systems, such as flow lines with closed loops or several product types. In addition, it is desirable to develop a model which allows for simultaneous optimization of the parameters for the order policy and the buffer capacities at any station of the line.

**Acknowledgments:** This research was supported in part by the Julius-Paul-Stiegler-Gedächtnis-Stiftung.

# A. Detailed results for Erlang-k and Cox-2 distributed instances

Table A.1.: 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 A.2.: Detailed results[1](Erlang-k distribution, $S = 7$)

| Sample | SCV | Optimal allocation | Max. dev. from $TH^*$ (%) | Initial bounds $i=2$ | | | | | | $i=3$ | | | | | $i=4$ | | | | $i=5$ | | | $i=6$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ | $b_6$ | $\sum_{j=1}^{2} b_j$ | $\sum_{j=2}^{3} b_j$ | $\sum_{j=3}^{4} b_j$ | $\sum_{j=4}^{5} b_j$ | $\sum_{j=5}^{6} b_j$ | $\sum_{j=1}^{3} b_j$ | $\sum_{j=2}^{4} b_j$ | $\sum_{j=3}^{5} b_j$ | $\sum_{j=4}^{6} b_j$ | $\sum_{j=1}^{4} b_j$ | $\sum_{j=2}^{5} b_j$ | $\sum_{j=3}^{6} b_j$ | $\sum_{j=1}^{5} b_j$ | $\sum_{j=2}^{6} b_j$ |
| 1,4,10 | 0.25 | 1,2,2,3,1,1 | 0.21 | **1** | 1 | 1 | 1 | **1** | **1** | 2 | 3 | 3 | 3 | **2** | 4 | 5 | 5 | 4 | 6 | 6 | 6 | 8 | 8 |
| 2,5,7 | 0.25 | 1,2,2,2,1,2 | 0.14 | **1** | 1 | 1 | 1 | **1** | 1 | 2 | 3 | 3 | **3** | 2 | 4 | 5 | **5** | 4 | 6 | 6 | 7 | **8** | 8 |
| 3,6 | 0.25 | 1,1,3,3,1,1 | −0.05 | **1** | **1** | 1 | 1 | **1** | **1** | **2** | 3 | 3 | 3 | **2** | 4 | 5 | 5 | 4 | 6 | 6 | 6 | 8 | 8 |
| 8,9 | 0.25 | 1,1,3,2,2,1 | 0.26 | **1** | **1** | 1 | 1 | 1 | **1** | **2** | 3 | 3 | 3 | 2 | 4 | 5 | 5 | 4 | **7** | 6 | 7 | 8 | 8 |
| 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 | 9 | 10 | 10 | 9 | 14 | 14 | 14 | 18 | 18 |
| 2 | 0.5 | 2,5,5,4,3,3 | −0.04 | 1 | 1 | 2 | 2 | 1 | 1 | 4 | 6 | 6 | 5 | 4 | 9 | 10 | 10 | 9 | 14 | 14 | 14 | 18 | 18 |
| 5 | 0.5 | 4,3,4,5,4,2 | −0.09 | 1 | 1 | 2 | 2 | 1 | 1 | 4 | 5 | 6 | 5 | 4 | 9 | 10 | 10 | 9 | 14 | 14 | 14 | 18 | 18 |
| 6 | 0.5 | 2,4,5,5,4,2 | 0.03 | 1 | 1 | 2 | 2 | 1 | 1 | 4 | 6 | 6 | 6 | 4 | 9 | 10 | 10 | 9 | 14 | 14 | 14 | 18 | 18 |

[1]Tight bounds are marked in bold

Table A.3.: Detailed results (Cox-2 distribution, $S = 7$)

| Sample | SCV | Optimal allocation | Max. dev. from $TH^*$ (%) | Initial bounds | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $i=2$ | | | | | | $i=3$ | | | | | $i=4$ | | | | $i=5$ | | | | $i=6$ | |
| | | | | $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ | $b_6$ | $\sum_{j=1}^{2} b_j$ | $\sum_{j=2}^{3} b_j$ | $\sum_{j=3}^{4} b_j$ | $\sum_{j=4}^{5} b_j$ | $\sum_{j=5}^{6} b_j$ | $\sum_{j=1}^{3} b_j$ | $\sum_{j=2}^{4} b_j$ | $\sum_{j=3}^{5} b_j$ | $\sum_{j=4}^{6} b_j$ | $\sum_{j=1}^{4} b_j$ | $\sum_{j=2}^{5} b_j$ | $\sum_{j=3}^{6} b_j$ | $\sum_{j=1}^{5} b_j$ | $\sum_{j=2}^{6} b_j$ |
| 1 | 1.0 | 6,7,10,12,7,5 | 0.11 | 3 | 3 | 5 | 5 | 3 | 3 | 9 | 12 | 13 | 12 | 9 | 19 | 21 | 21 | 20 | 29 | 30 | 30 | 38 | 38 |
| 2 | 1.0 | 7,7,8,10,9,6 | 0.05 | 3 | 3 | 5 | 5 | 3 | 3 | 9 | 12 | 13 | 12 | 9 | 20 | 21 | 21 | 20 | 29 | 29 | 29 | 38 | 38 |
| 3 | 1.0 | 6,7,10,10,7,6 | 0.03 | 3 | 3 | 5 | 5 | 3 | 3 | 9 | 12 | 13 | 12 | 9 | 20 | 21 | 21 | 19 | 29 | 30 | 29 | 38 | 38 |
| 4 | 1.0 | 6,7,10,10,7,6 | 0.03 | 3 | 3 | 5 | 5 | 3 | 3 | 9 | 12 | 13 | 12 | 9 | 20 | 21 | 21 | 19 | 29 | 29 | 29 | 38 | 38 |
| 5 | 1.0 | 6,7,11,9,7,6 | −0.04 | 3 | 3 | 5 | 5 | 3 | 3 | 9 | 12 | 13 | 12 | 9 | 19 | 21 | 21 | 20 | 29 | 29 | 29 | 38 | 38 |
| 6 | 1.0 | 7,8,8,9,8,7 | 0.06 | 3 | 3 | 5 | 5 | 3 | 3 | 9 | 12 | 13 | 12 | 9 | 19 | 21 | 21 | 20 | 29 | 30 | 29 | 38 | 38 |
| 7 | 1.0 | 6,7,11,9,7,6 | −0.04 | 3 | 3 | 5 | 5 | 3 | 3 | 9 | 12 | 13 | 12 | 9 | 20 | 21 | 21 | 19 | 29 | 29 | 29 | 38 | 38 |
| 8 | 1.0 | 6,8,9,10,7,6 | 0.01 | 3 | 3 | 5 | 5 | 3 | 3 | 9 | 12 | 13 | 12 | 9 | 19 | 21 | 21 | 20 | 29 | 29 | 29 | 38 | 38 |
| 9 | 1.0 | 5,8,10,10,6,8 | 0.04 | 3 | 3 | 5 | 5 | 3 | 3 | 9 | 12 | 13 | 12 | 9 | 19 | 21 | 21 | 20 | 29 | 30 | 29 | 38 | 38 |
| 10 | 1.0 | 6,7,10,9,7,7 | −0.04 | 3 | 3 | 5 | 5 | 3 | 3 | 9 | 12 | 13 | 12 | 9 | 19 | 21 | 21 | 19 | 29 | 29 | 29 | 38 | 38 |
| 1 | 2.0 | 13,18,18,19,16,13 | −0.13 | 5 | 5 | 9 | 9 | 5 | 5 | 18 | 24 | 26 | 24 | 18 | 40 | 43 | 43 | 41 | 60 | 61 | 61 | 79 | 78 |
| 2 | 2.0 | 13,16,19,20,15,14 | −0.09 | 6 | 5 | 9 | 9 | 5 | 6 | 18 | 24 | 26 | 24 | 18 | 40 | 43 | 43 | 41 | 60 | 61 | 61 | 78 | 79 |
| 3 | 2.0 | 13,16,19,20,17,13 | 0.07 | 6 | 6 | 9 | 9 | 6 | 6 | 18 | 25 | 26 | 25 | 18 | 41 | 43 | 44 | 41 | 61 | 62 | 62 | 80 | 80 |
| 4 | 2.0 | 12,18,20,19,14,14 | −0.15 | 5 | 6 | 9 | 9 | 5 | 5 | 18 | 24 | 26 | 24 | 18 | 41 | 44 | 43 | 40 | 61 | 61 | 61 | 79 | 79 |
| 5 | 2.0 | 13,15,20,19,16,13 | −0.20 | 6 | 5 | 9 | 9 | 5 | 5 | 18 | 24 | 25 | 24 | 18 | 40 | 42 | 43 | 40 | 60 | 61 | 60 | 79 | 78 |
| 6 | 2.0 | 12,16,19,20,18,13 | 0.03 | 5 | 6 | 9 | 9 | 5 | 5 | 18 | 25 | 27 | 25 | 18 | 41 | 44 | 44 | 41 | 61 | 62 | 62 | 80 | 80 |
| 7 | 2.0 | 13,15,19,19,16,13 | −0.36 | 5 | 5 | 9 | 9 | 5 | 5 | 18 | 23 | 26 | 24 | 18 | 39 | 43 | 43 | 40 | 60 | 60 | 60 | 78 | 78 |
| 8 | 2.0 | 11,17,20,20,16,13 | −0.09 | 5 | 6 | 9 | 9 | 5 | 6 | 18 | 24 | 26 | 24 | 19 | 40 | 43 | 43 | 41 | 60 | 61 | 61 | 79 | 80 |
| 9 | 2.0 | 12,16,20,20,16,13 | −0.05 | 5 | 5 | 9 | 9 | 5 | 5 | 18 | 24 | 26 | 24 | 18 | 40 | 43 | 43 | 41 | 60 | 61 | 61 | 79 | 79 |
| 10 | 2.0 | 12,17,20,19,16,13 | −0.07 | 6 | 5 | 9 | 9 | 6 | 6 | 18 | 24 | 26 | 24 | 18 | 40 | 43 | 43 | 41 | 61 | 61 | 61 | 79 | 79 |

# B. Sample-based evaluation algorithms for lines with limited supply

Algorithms 3 and 4 present the evaluation for flow lines with (s,q)-order policies and (r,S)-order policies, respectively. Table B.1 contains the notation.

Table B.1.: Notation for the throughput evaluation

| | |
|---|---|
| **Indices** | |
| $w = 1, \ldots, W$ | Workpieces |
| $k = 1, \ldots, \lceil \frac{W}{q} \rceil$ | Orders in the (s,q)-policy |
| $l$ | Control variable for the review intervals |
| $k$ | Control variable for the orders in the (r,S)-policy |
| **Parameters** | |
| $s$ | Reorder point |
| $q$ | Order quantity |
| $r$ | Review interval |
| $S$ | Order-up-to level |
| $T$ | Lead time of an order |
| $d_{m,w}$ | Sampled effective processing time of workpiece $w$ at station $m$ |
| $W_0$ | Number of workpieces in the warm-up |
| **Decision variables** | |
| $XS_{m,w}$ | Start time of workpiece $w$ at station $m$ |
| $XF_{m,w}$ | Departure time of workpiece $w$ from station $m$ |
| $\text{ArrivalTime}_k$ | Arrival time of order $k$ |
| $\text{Quantity}_k$ | Order quantity of order $k$ |
| **Performance measures** | |
| $E[TH(X_1, \ldots, X_{M-1})]$ | Throughput resulting from allocation $X_1, \ldots, X_{M-1}$ |

1: $XS_{1,1} = T$
2: **for** all stations $m < M$ **do**
3: $\quad XF_{m,1} = XS_{m,1} + d_{m,1}$
4: $\quad XS_{m+1,1} = XF_{m,1}$
5: **end for**
6: $XF_{M,1} = XS_{M,1} + d_{M,1}$
7: **for** all workpieces $w > 1$ **do**
8: $\quad$ **for** all stations $m < M$ **do**
9: $\quad\quad$ **if** $m = 1$ **then**
10: $\quad\quad\quad i = \lceil \frac{s}{q} \rceil$
11: $\quad\quad\quad$ **if** $w \leq i \cdot q$ **then**
12: $\quad\quad\quad\quad XS_{1,w} = \max\{XF_{1,w-1}, T\}$
13: $\quad\quad\quad$ **else**
14: $\quad\quad\quad\quad k = \lceil \frac{w}{q} \rceil$
15: $\quad\quad\quad\quad XS_{1,w} = \max\{XF_{1,w-1}, XS_{1,(k-1)\cdot q-s} + T\}$
16: $\quad\quad\quad$ **end if**
17: $\quad\quad$ **else**
18: $\quad\quad\quad XS_{m,w} = \max\{XF_{m,w-1}, XF_{m-1,w}\}$
19: $\quad\quad$ **end if**
20: $\quad\quad$ **if** $X_m = 0$ **then**
21: $\quad\quad\quad XF_{m,w} = \max\{XS_{m,w} + d_{m,w}, XF_{m+1,w-1}\}$
22: $\quad\quad$ **else if** $X_m \geq w$ **then**
23: $\quad\quad\quad XF_{m,w} = XS_{m,w} + d_{m,w}$
24: $\quad\quad$ **else**
25: $\quad\quad\quad XF_{m,w} = \max\{XS_{m,w} + d_{m,w}, XS_{m+1,w-X_m}\}$
26: $\quad\quad$ **end if**
27: $\quad$ **end for**
28: $\quad XS_{M,w} = \max\{XF_{M,w-1}, XF_{M-1,w}\}$
29: $\quad XF_{M,w} = XS_{M,w} + d_{M,w}$
30: **end for**
31: $E[TH(X_1, \ldots, X_{M-1})] = \frac{W - W_0}{XF_{M,W} - XF_{M,W_0}}$

Algorithm 3: Sample-based throughput evaluation with an (s,q)-order policy

$l = 1$, $k = 1$
InventoryPosition $= \lceil \frac{S}{q} \rceil \cdot q$
InventoryLevel $= \lceil \frac{S}{q} \rceil \cdot q$
ArrivalTime$_1 = T$
Quantity$_1 = \lceil \frac{S}{q} \rceil \cdot q$
$XS_{1,1} = T$
**for** all stations $m < M$ **do**
    $XF_{m,1} = XS_{m,1} + d_{m,1}$
    $XS_{m+1,1} = XF_{m,1}$
**end for**
$XF_{M,1} = XS_{M,1} + d_{M,1}$
**for** all workpieces $w > 1$ **do**
    **for** all stations $m < M$ **do**
        **if** $m = 1$ **then**
            **if** InventoryLevel $> 0$ **then**
                $XS_{1,w} = XF_{1,w-1}$
                InventoryPosition $-= 1$
                InventoryLevel $-= 1$
            **else**
                $XS_{1,w} = \max\{XF_{1,w-1}, \text{ArrivalTime}_k\}$
                InventoryLevel $+=$ Quantity$_k - 1$
                $k+= 1$
            **end if**
        **else**
            $XS_{m,w} = \max\{XF_{m,w-1}, XF_{m-1,w}\}$
        **end if**

Algorithm 4: Sample-based throughput evaluation with an (r,S)-order policy

**if** $X_m = 0$ **then**
  $XF_{m,w} = \max\{XS_{m,w} + d_{m,w}, XF_{m+1,w-1}\}$
**else if** $X_m \geq w$ **then**
  $XF_{m,w} = XS_{m,w} + d_{m,w}$
**else**
  $XF_{m,w} = \max\{XS_{m,w} + d_{m,w}, XS_{m+1,w-X_m}\}$
**end if**
**if** $m = 1$ **then**
  **if** $l \cdot r \leq XF_{1,w}$ and InventoryPosition $< S$ **then**
    ArrivalTime.add($l \cdot r + T$)
    Quantity.add($\lceil \frac{S-\text{InventoryPosition}}{q} \rceil \cdot q$)
    InventoryPosition$+= \lceil \frac{S-\text{InventoryPosition}}{q} \rceil \cdot q$
    $l+=1$
  **end if**
  **if** ArrivalTime$_k \leq XF_{1,w}$ **then**
    InventoryLevel$+=$ Quantity$_k$
    $k+=1$
  **end if**
**end if**
**end for**
$XS_{M,w} = \max\{XF_{M,w-1}, XF_{M-1,w}\}$
$XF_{M,w} = XS_{M,w} + d_{M,w}$
**end for**
$E[TH(X_1, \ldots, X_{M-1})] = \frac{W - W_0}{XF_{M,W} - XF_{M,W_0}}$

Sample-based throughput evaluation with an (r,S)-order policy (continued)

# Bibliography


Alden, J. M., L. D. Burns, T. Costy, R. D. Hutton, C. A. Jackson, D. S. Kim, K. A. Kohls, J. H. Owen, M. A. Turnquist, and D. J. V. Veen (2006). General Motors increases its production throughput. *Interfaces 36*(1), 6–25.

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

Alfieri, A. and A. Matta (2013). Mathematical programming time-based decomposition algorithm for discrete event simulation. *European Journal of Operational Research 231*(3), 557–566.

Alfieri, A., A. Matta, and G. Pedrielli (2015). Mathematical programming models for joint simulation-optimization applied to closed queueing networks. *Annals of Operations Research 231*(1), 105–127.

Alon, G., D. P. Kroese, T. Raviv, and R. Y. Rubinstein (2005). Application of the cross-entropy method to the buffer allocation problem in a simulation-based environment. *Annals of Operations Research 134*(1), 137–151.

Axsäter, S. and K. Rosling (1993). Installation vs. echelon stock policies for multilevel inventory control. *Management Science 39*(10), 1274–1280.

Bai, L. and P. A. Rubin (2009). Combinatorial benders cuts for the minimum tollbooth problem. *Operations Research 57*(6), 1510–1522.

Bekker, J. (2013). Multi-objective buffer space allocation with the cross-entropy method. *International Journal of Simulation Modelling 12*(1), 50–61.

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

Burman, M. H., S. B. Gershwin, and C. Suyematsu (1998). Hewlett-Packard uses operations research to improve the design of a printer production line. *Interfaces 28*(1), 24–36.

Buzacott, J. A. (1971). The role of inventory banks in flow-line production systems. *International Journal of Production Research 9*(4), 425–436.

Buzacott, J. A. and J. G. Shanthikumar (1993). *Stochastic models of manufacturing systems* (4 ed.). Englewood Cliffs, NJ, USA: Prentice Hall.

Caramanis, M. (1987). Production system design: A discrete event dynamic system and generalized benders' decomposition approach. *International Journal of Production Research 25*(8), 1223–1234.

Chan, W. K. V. and L. Schruben (2008). Optimization models of discrete-event system dynamics. *Operations Research 56*(5), 1218–1237.

Chen, J. C. and L. Chen (1993). A fast simulator for tandem queueing systems. *Computers and Industrial Engineering 24*(2), 267–280.

Chiang, S.-Y., C.-T. Kuo, and S. M. Meerkov (2000). DT-bottlenecks in serial production lines: Theory and application. *IEEE Transactions on Robotics and Automation 16*(5), 567–580.

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

Colledani, M., M. Ekvall, T. Lundholm, P. Moriggi, A. Polato, and T. Tolio (2010). Analytical methods to support continuous improvements at Scania. *International Journal of Production Research 48*(7), 1913–1945.

Cooke, R. M., A. Bosma, and F. Härte (2005). A practical model of Heineken's bottle filling line with dependent failures. *European Journal of Operational Research 164*(2), 491–504.

Dallery, Y. and S. B. Gershwin (1992). Manufacturing flow line systems: a review of models and analytical results. *Queueing Systems 12*(1-2), 3–94.

Demir, L., S. Tunali, and D. T. Eliiyi (2014). The state of the art on buffer allocation problem: A comprehensive survey. *Journal of Intelligent Manufacturing 25*(3), 371–392.

Demir, L., S. Tunali, and A. Løkketangen (2011). A tabu search approach for buffer allocation in production lines with unreliable machines. *Engineering Optimization 43*(2), 213–231.

Diamantidis, A. C. and C. T. Papadopoulos (2004). A dynamic programming algorithm for the buffer allocation problem in homogeneous asymptotically reliable serial production lines. *Mathematical Problems in Engineering 3*, 209–223.

Dolgui, A., A. Eremeev, A. Kolokolov, and V. Sigaev (2002). A genetic algorithm for the allocation of buffer storage capacities in a production line with unreliable machines. *Journal of Mathematical Modelling and Algorithms 1*(2), 89–104.

Dolgui, A., A. Eremeev, M. Y. Kovalyov, and V. Sigaev (2013). Complexity of buffer capacity allocation problems for production lines with unreliable machines. *Journal of Mathematical Modelling and Algorithms in Operations Research 12*(2), 155–165.

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

Enginarlar, E., J. Li, and S. M. Meerkov (2005). How lean can lean buffers be? *IIE Transactions 37*(4), 333–342.

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

Gürkan, G. (2000). Simulation optimization of buffer allocations in production lines with unreliable machines. *Annals of Operations Research 93*(1-4), 177–216.

Han, M.-S. and D.-J. Park (2002). Optimal buffer allocation of serial production lines with quality inspection machines. *Computers and Industrial Engineering 42*(1), 75–89.

Helber, S. (2001). Cash-flow-oriented buffer allocation in stochastic flow lines. *International Journal of Production Research 39*(14), 3061–3083.

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

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.

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

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.

Kim, S. and H.-J. Lee (2001). Allocation of buffer capacity to minimize average work-in-process. *Production Planning and Control 12*(7), 706–716.

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

Lee, H.-T., S.-K. Chen, and S. Chang (2009). A meta-heuristic approach to buffer allocation in production line. *Journal of C.C.I.T 38*(1), 167–178.

Lee, S.-D. and S.-H. Ho (2002). Buffer sizing in manufacturing production systems with complex routings. *International Journal of Computer Integrated Manufacturing 15*(5), 440–452.

Levantesi, R., A. Matta, and T. Tolio (2001). A new algorithm for buffer allocation in production lines. In *Proceedings of the Third 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. *International Journal of Production Research 51*(23-24), 7235–7249.

Li, J. and S. M. Meerkov (2009). *Production Systems Engineering*. Boston, MA, USA: Springer Science+ Business Media, LLC.

Liberopoulos, G. and P. Tsarouhas (2002). Systems analysis speeds up Chipita's food-processing line. *Interfaces 32*(3), 62–76.

Liberopoulos, G. and P. Tsarouhas (2005). Reliability analysis of an automated pizza production line. *Journal of Food Engineering 69*(1), 79–96.

Lutz, C. M., K. R. Davis, and M. Sun (1998). Determining buffer location and size in production lines using tabu search. *European Journal of Operational Research 106*(2), 301–316.

Massim, Y., F. Yalaoui, L. Amodeo, E. Chatelet, and A. Zeblah (2010). Efficient combined immune-decomposition algorithm for optimal buffer allocation in production lines for throughput and profit maximization. *Computers and Operations Research 37*(4), 611–620.

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

Matta, A. and R. Chefson (2005). Formal properties of closed flow lines with limited buffer capacities and random processing times. In *Proceedings of the European Simulation and Modelling Conference*, Porto, Portugal, pp. 190–194.

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 2014 Winter Simulation Conference*, Savannah, GA, USA, pp. 3983–3994.

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

Nahas, N., D. Ait-Kadi, and M. Nourelfath (2006). A new approach for buffer allocation in unreliable production lines. *International Journal of Production Economics 103*(2), 873–881.

Papadopoulos, H. T. and M. I. Vidalis (2001). A heuristic algorithm for the buffer allocation in unreliable unbalanced production lines. *Computers and Industrial Engineering 41*(3), 261–277.

Patchong, A., T. Lemoine, and G. Kern (2003). Improving car body production at PSA Peugeot Citroen. *Interfaces 33*(1), 36–49.

Pedrielli, G., A. Alfieri, and A. Matta (2015). Integrated simulation–optimisation of pull control systems. *International Journal of Production Research 53*(14), 4317–4336.

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

Sabuncuoglu, I., E. Erel, and Y. Gocgun (2006). Analysis of serial production lines: Characterisation study and a new heuristic procedure for optimal buffer allocation. *International Journal of Production Research 44*(13), 2499–2523.

Saliby, E. (1990a). Descriptive sampling: A better approach to monte carlo simulation. *The Journal of the Operational Research Society 41*(12), 1133–1142.

Saliby, E. (1990b). Understanding the variability of simulation results: An empirical study. *The Journal of the Operational Research Society 41*(4), 319–327.

Savsar, M. (2006). Buffer allocation in serial production lines with preventive and corrective maintenance operations. *Kuwait Journal of Science and Engineering 33*(2), 253–266.

Schruben, L. W. (2000). Mathematical programming models of discrete event system dynamics. In *Proceedings of the 32nd conference on Winter simulation*, Orlando, FL, USA, pp. 381–385.

Shi, C. and S. B. Gershwin (2009). An efficient buffer design algorithm for production line profit maximization. *International Journal of Production Economics 122*(2), 725–740.

Shi, C. and S. B. Gershwin (2014). A segmentation approach for solving buffer allocation problems in large production systems. *International Journal of Production Research* (In Press), 1–21.

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

Silver, E. A., D. F. Pyke, and R. Peterson (1998). *Inventory management and production planning and scheduling* (3 ed.). New York, NY, USA: John Wiley.

Smith, J. M. and F. R. B. Cruz (2005). The buffer allocation problem for general finite buffer queueing networks. *IIE Transactions 37*(4), 343–365.

Spinellis, D. D. and C. T. Papadopoulos (2000). A simulated annealing approach for buffer allocation in reliable production lines. *Annals of Operations Research 93*(1-4), 373–384.

Stolletz, R. and S. Weiss (2013). Buffer allocation using exact linear programming formulations and sampling approaches. In *Manufacturing Modelling, Management, and Control*, Volume 7(1), St. Petersburg, Russia, pp. 1435–1440.

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

Vergara, H. A. and D. S. Kim (2009). A new method for the placement of buffers in serial production lines. *International Journal of Production Research 47*(16), 4437–4456.

Vouros, G. A. and H. T. Papadopoulos (1998). Buffer allocation in unreliable production lines using a knowledge based system. *Computers and Operations Research 25*(12), 1055–1067.

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.

Yamashita, H. and T. Altiok (1998). Buffer capacity allocation for a desired throughput in production lines. *IIE Transactions 30*(10), 883–891.

# Curriculum vitae

**Professional experience**

| | |
|---|---|
| 06/2011 - present | Research Assistant, Chair of Production Management, University of Mannheim |
| 03/2014 | Visiting Researcher, School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, China |

**Education**

| | |
|---|---|
| 01/2010 - 08/2010 | Exchange semester, Universitat de València |
| 10/2006 - 09/2010 | Business Mathematics (Diploma), University of Cologne |
| 06/2006 | Abitur, Paul-Klee-Gymnasium, Overath |