

TYPE-II STOCHASTIC TRANSFER LINE BALANCING PROBLEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

YASIN ERSIN TELEMECI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
INDUSTRIAL ENGINEERING

FEBRUARY 2026

Approval of the thesis:

TYPE-II STOCHASTIC TRANSFER LINE BALANCING PROBLEM

submitted by **YASIN ERSİN TELEMECİ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Industrial Engineering, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Zeynep Pelin Bayındır
Head of the Department, **Industrial Engineering** _____

Prof. Dr. Meral Azizoğlu
Supervisor, **Industrial Engineering, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Mustafa Kemal Tural
Industrial Engineering, METU _____

Prof. Dr. Meral Azizoğlu
Industrial Engineering, METU _____

Assoc. Prof. Dr. Serhat Gül
Industrial Engineering, TEDU _____

Assoc. Prof. Dr. Mehmet Rüştü Taner
Industrial Engineering, TEDU _____

Assist. Prof. Dr. Nader Ghaffarinasab
Industrial Engineering, METU _____

Date: 20.02.2026

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Last name: Yasin Ersin Telemeci

Signature:

ABSTRACT

TYPE-II STOCHASTIC TRANSFER LINE BALANCING PROBLEM

Telemeci, Yasin Ersin
Doctor of Philosophy, Industrial Engineering
Supervisor: Prof. Dr. Meral Azizoğlu

February 2026, 123 pages

This thesis studies the Type-II Transfer Line Balancing Problem under stochastic workstation maintenance times. In contrast to deterministic formulations that incorporate maintenance effects through average availability parameters, maintenance durations are explicitly modeled as random variables, and their impact is directly embedded in cycle time performance.

To capture maintenance-induced variability, a scenario-based stochastic programming formulation, a constraint programming model, and a Branch and Bound algorithm are developed with the objective of minimizing expected cycle time subject to technological, precedence, and workstation exclusion and inclusion constraints. Due to the combinatorial structure of block formation and operation assignment decisions, exact solution approaches become computationally challenging for large-sized instances. To address large-sized instances, matheuristic solution approaches are introduced that integrate mathematical programming components with structured improvement procedures. These approaches employ decomposition-based block formation, workstation assignment mechanisms, and local improvement procedures to enhance solution quality while maintaining computational efficiency.

The results of extensive computational experiments reveal that exact methods are effective for small-to-medium-sized instances, while the proposed matheuristic procedures can solve large-sized instances within reasonable computational times and produce high-quality solutions.

Keywords: Type-II Transfer Line Balancing, Stochastic Programming, Branch and Bound Algorithm, Matheuristic

ÖZ

TİP-2 STOKASTİK TRANSFER HATTI Dengeleme Problemi

Telemeci, Yasin Ersin
Doktora, Endüstri Mühendisliği
Tez Yöneticisi: Prof. Dr. Meral Azizoğlu

Şubat 2026, 123 sayfa

Bu tez, stokastik iş istasyonu bakım sürelerini göz önünde bulundurarak Tip-II Transfer Hattı Dengeleme Problemini incelemektedir. Bakım sürelerinin etkilerini ortalama parametreler aracılığıyla dahil eden deterministik formülasyonların aksine, bu çalışmada bakım süreleri açıkça rassal değişkenler olarak modellenmiş ve bunların etkileri doğrudan çevrim süresi performansına entegre edilmiştir.

Bakım kaynaklı değişkenliği ele almak amacıyla; teknolojik, öncelik ve iş istasyonu dışlama ve dahil etme kısıtlarına tabi olarak beklenen çevrim süresini en aza indirme hedefiyle senaryo tabanlı bir stokastik programlama formülasyonu, bir kısıt programlama modeli ve bir Dal ve Sınır algoritması geliştirilmiştir.

Blok oluşturma ve operasyon atama kararlarının kombinatoriyal yapısı nedeniyle, kesin çözüm yaklaşımları büyük boyutlu problemler için hesaplama açısından zorlayıcı hale gelmektedir. Büyük boyutlu problemleri çözebilmek adına, matematiksel programlama bileşenleri yapılandırılmış iyileştirme prosedürleri ile bütünleştirilmiş matematiksel sezgisel çözüm yaklaşımları sunulmuştur. Bu yaklaşımlar, hesaplama verimliliğini korurken çözüm kalitesini artırmak için ayrıştırma tabanlı blok oluşturma, iş istasyonu atama mekanizmaları ve yerel iyileştirme prosedürleri kullanmaktadır.

Kapsamlı hesaplama deneylerinin sonuçları, kesin yöntemlerin küçük ve orta ölçekli problemler için etkili olduğunu; önerilen matematiksel sezgisel prosedürlerin ise büyük ölçekli problemleri makul hesaplama süreleri içinde çözebildiğini ve yüksek kaliteli çözümler ürettiğini ortaya koymaktadır.

Anahtar Kelimeler: Tip-2 Transfer Hattı Dengeleme, Stokastik Programlama, Dal ve Sınır Algoritması, Matematiksel Sezgisel

To my mother...

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Prof. Dr. Meral Azizođlu. Her continuous guidance, insightful contributions, and patience have been instrumental to this study. I consider myself exceptionally fortunate to have her as a mentor throughout my academic life.

I want to thank my company ASELSAN INC. and my managers for allowing me to spare time for this study.

A very special thank you goes to my dear friend, iya Aydođan, for their unwavering support and encouragement throughout this journey. Thank you for the endless conversations and the much-needed distractions. Having a friend like you in my corner has been one of the greatest blessings of my life.

The author is financially supported by the Scientific and Technological Research Council of Trkiye (TBİTAK) under the BİDEB 2211-A National PhD Scholarship Program.

TABLE OF CONTENTS

| | |
|---|------|
| ABSTRACT | v |
| ÖZ..... | vii |
| ACKNOWLEDGMENTS..... | x |
| TABLE OF CONTENTS | xi |
| LIST OF TABLES | xiii |
| LIST OF FIGURES..... | xvi |
| INTRODUCTION..... | 1 |
| LITERATURE REVIEW..... | 5 |
| 2.1 Literature on Stochastic Assembly Line Problem..... | 6 |
| 2.2 Literature on Transfer Line Balancing Problem | 9 |
| 2.3 Literature on Reconfigurable Transfer Line Balancing Problem..... | 12 |
| 2.4 Contribution of the Thesis..... | 15 |
| STOCHASTIC TRANSFER LINE BALANCING PROBLEM | 17 |
| 3.1 Problem Statement | 17 |
| 3.2 Brief Background on Stochastic Programming..... | 19 |
| 3.3 Stochastic Transfer Line Balancing Problem..... | 20 |
| 3.4 Finding the Earliest and Latest Workstation Values:..... | 23 |
| 3.5 Stochastic Programming Model | 28 |
| 3.6 Constraint Programming Model..... | 36 |
| 3.7 Expected Value Problem and the Value of the Stochastic Solution | 39 |

| | |
|---|-----|
| BRANCH AND BOUND ALGORITHM..... | 43 |
| 4.1 Branching Scheme | 43 |
| 4.2 Lower Bounds..... | 46 |
| HEURISTIC APPROACHES..... | 51 |
| 5.1 Alpha Percent Heuristic – Branch and Bound-based | 51 |
| 5.2 Matheuristics – Mathematical Model-based Heuristics | 52 |
| 5.3 Rule-based Heuristic..... | 66 |
| COMPUTATIONAL EXPERIMENT | 75 |
| 6.1 Data Generation | 75 |
| 6.2 Performance Measures..... | 79 |
| 6.3 Preliminary Experiments | 83 |
| 6.4 Computational Experiments | 98 |
| CONCLUSION..... | 117 |
| REFERENCES | 119 |
| CURRICULUM VITAE..... | 123 |

LIST OF TABLES

| | |
|---|----|
| Table 1: Literature on Stochastic Assembly Line Balancing Problem | 8 |
| Table 2: Literature on Transfer Line Balancing Problem | 11 |
| Table 3: Literature on Reconfigurable Transfer Line Balancing Problem | 14 |
| Table 4: Case 1 scenarios and related probabilities | 22 |
| Table 5: Case 2 scenarios and related probabilities | 23 |
| Table 6: Operations – descriptions and times | 32 |
| Table 7: Operation times and technological constraints | 33 |
| Table 8: Independent maintenance times and probabilities | 33 |
| Table 9: Scenarios and maintenance times with related probabilities | 33 |
| Table 10: Ordered operation times and technological constraints | 34 |
| Table 11: E_i and L_i values | 35 |
| Table 12: Assignments of operations and block times | 35 |
| Table 13: Cycle time of scenarios with probabilities..... | 36 |
| Table 14: Average maintenance times | 40 |
| Table 15: Assignments of operations and block times | 41 |
| Table 16: Cycle time for each scenario with probabilities..... | 41 |
| Table 17: Ordered operation times and technological constraints | 47 |
| Table 18: Operation assignments to groups | 48 |
| Table 19: Assignments of operations and block times | 49 |
| Table 20: Lower Bounds..... | 50 |
| Table 21: Assignments of operations to blocks Model B0 | 70 |
| Table 22: Assignments of operations to blocks Model B1 | 70 |
| Table 23: Assignments of operations to blocks Model B2 | 70 |
| Table 24: Assignments of operations to blocks Model B3 | 70 |
| Table 25: Assignments of blocks to workstations Model B0-W | 71 |
| Table 26: Assignments of blocks to workstations Model B1-W | 71 |
| Table 27: Assignments of blocks to workstations Model B2-W | 71 |

| | |
|--|----|
| Table 28: Assignments of blocks to workstations Model B3-W..... | 71 |
| Table 29: Improved solution of Model B0-W-IP1 | 71 |
| Table 30: Improved solution of Model B1-W-IP1 | 72 |
| Table 31: Improved solution of Model B2-W-IP1 | 72 |
| Table 32: Improved solution of Model B3-W-IP1 | 72 |
| Table 33: Improved solution of Model B0-W-IP2 | 72 |
| Table 34: Improved solution of Model B1-W-IP2 | 72 |
| Table 35: Improved solution of Model B2-W-IP2 | 72 |
| Table 36: Improved solution of Model B3-W-IP2 | 73 |
| Table 37: Assignments of operations to blocks Model BFR..... | 73 |
| Table 38: Assignments of blocks to workstations Model BFR-BAR | 73 |
| Table 39: Improved solution of Model BFR-BAR-IP1 | 73 |
| Table 40: Final results of Model- and Rule-based heuristics | 73 |
| Table 41: The effect of distribution of processing times of operations..... | 84 |
| Table 42: The effect of earliest and latest workstation information..... | 85 |
| Table 43: The results of B&B algorithm (With/Without LB)..... | 86 |
| Table 44: CPU times of Model-based heuristics | 88 |
| Table 45: Percent deviation from optimal of Model-based heuristics | 89 |
| Table 46: Frequency best and optimal solutions of Model-based heuristics..... | 90 |
| Table 47: CPU times of improvement procedure 1 on Model-based heuristics..... | 91 |
| Table 48: CPU times of improvement procedure 2 on Model-based heuristics..... | 92 |
| Table 49: Percent deviations of improvement procedure 1 on Model-based heuristics..... | 93 |
| Table 50: Percent deviations of improvement procedure 2 on Model-based heuristics..... | 94 |
| Table 51: CPU times of Model-based heuristics after improvement procedures.... | 95 |
| Table 52: Percent deviations of Model B2- and B3-based heuristics after improvement procedures | 96 |
| Table 53: Frequency best and optimal solutions of Model B2- and B3-based heuristics after improvement procedures..... | 97 |

| | |
|--|-----|
| Table 54: The results for the experiment | 99 |
| Table 55: SP and CP model results | 101 |
| Table 56: SP Model Results..... | 102 |
| Table 57: Percent Optimality Gap of SP model for 15min and 2 hours | 102 |
| Table 58: B&B Results; CPU times and percent deviations..... | 104 |
| Table 59: B&B and B&B-based heuristic results; CPU times and percent deviations | 106 |
| Table 60: The number of blocks and block sets and CPU times by B2 and B3 ... | 107 |
| Table 61: CPU Times of matheuristics and Rule-based heuristic | 109 |
| Table 62: Percent deviations of matheuristics and Rule-based heuristic..... | 111 |
| Table 63: Deviations from SP solution when optimality gap < 2% | 112 |
| Table 64: Relative percent deviations when optimality gap > 2% | 114 |
| Table 65: Relative percent deviations of matheuristics when optimality gap > 2% | 115 |

LIST OF FIGURES

Figure 1: A transfer line configuration 1
Figure 2: Transfer Line..... 17
Figure 3: Body of a Mechanical Desktop (excerpted from Belmokhtar et al. 2006)
..... 31
Figure 4: Connected subgraph of operations 48

CHAPTER 1

INTRODUCTION

Transfer lines are widely employed in manufacturing environments that require the production of large volumes of standardized, high-quality products. These systems consist of a sequence of unit-head machines, where each machine can be equipped with multiple spindle heads (blocks), and each block performs several operations simultaneously using multiple spindles. Determining the number of machines (workstations), spindle heads (blocks per workstation), and spindles (operations per block) constitutes a critical design challenge, as these decisions directly affect the overall investment cost. Since transfer lines require substantial investment in machinery and material handling systems, their design must ensure high productivity and efficient utilization to justify the associated costs.

Figure 1 is a pictorial representation of the transfer lines.

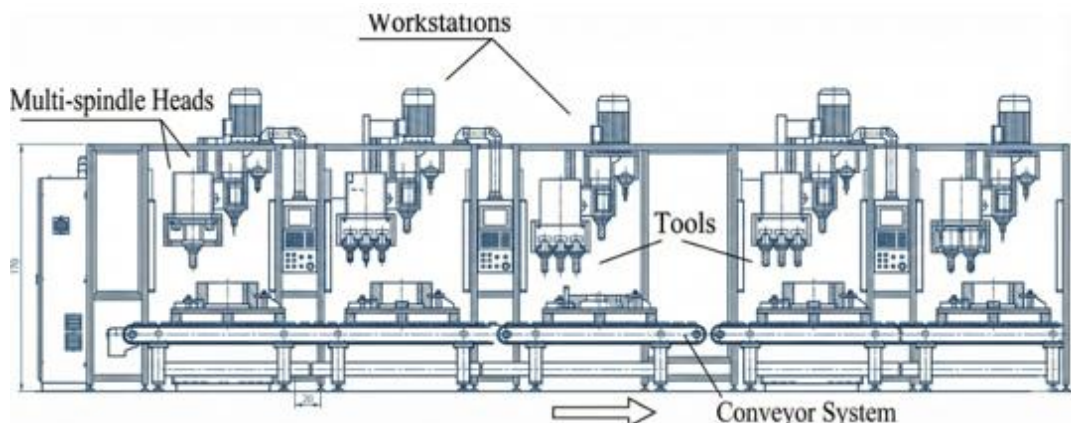


Figure 1: A transfer line configuration

In such systems, operations are grouped into blocks, processed in parallel, and sequentially transferred between workstations through automated handling mechanisms. Once all blocks at a workstation complete their assigned operations,

the part moves to the next workstation. This highly structured and synchronized production mode makes transfer lines extremely productive but also highly sensitive to disturbances.

The problem of efficiently assigning operations to blocks and blocks to workstations, while respecting technological and precedence constraints, is known as the Transfer Line Balancing Problem (TLBP). The design phase, known as Type-I TLBP, focuses on determining the optimal number of workstations, blocks per workstation, and operations per block to minimize the overall investment cost for a given cycle time. In contrast, Type-II TLBP addresses the operational phase once the transfer line is installed, where the primary objective is to minimize the cycle time while ensuring feasibility with respect to system constraints. Due to the combinatorial nature of operation assignments and the presence of block-level parallelism, TLBP is significantly more complex than classical assembly line balancing problems.

Most early studies on the TLBP assume deterministic system parameters, including operation times, machine availability, and maintenance activities. Under such assumptions, line design and balancing decisions are made based on fixed and known values. However, modern manufacturing environments are characterized by increasing product variety, shorter product life cycles, intensified global competition, and frequent changes in market demand. These conditions challenge the validity of purely deterministic design approaches and highlight the need for models that capture the variability inherent in real production systems.

In practice, transfer lines are subject to multiple sources of uncertainty that significantly affect their performance. Variability in operation times, machine reliability, and maintenance activities can lead to deviations from planned cycle times, reduced throughput, and increased downtime. Among these sources, workstation maintenance activities—such as cleaning, lubrication, calibration, minor repairs, and component replacement—play a critical role in determining

effective workstation availability. Maintenance durations are inherently variable and difficult to predict accurately, yet they are commonly treated as deterministic or implicitly incorporated into average availability measures in existing line design models. Ignoring this variability at the design stage may result in underestimated cycle times and inefficient balancing decisions.

At the same time, reconfigurable manufacturing systems have emerged to address the need for adaptability in manufacturing design. Reconfigurable transfer lines accommodate changes in production requirements through modular structures that allow adjustments in system capacity and functionality. Beyond physical modification, reconfigurability also reflects a system's ability to maintain acceptable performance under varying operating conditions. From this perspective, incorporating uncertainty into balancing stage supports the development of more adaptive transfer line configurations.

Motivated by these observations, this thesis addresses the transfer line balancing problem by explicitly incorporating stochastic workstation maintenance times into the design of a transfer line. In contrast to existing studies where maintenance is absorbed into average availability parameters, maintenance durations are modeled as random variables with known probability distributions, allowing their variability to be directly embedded in balancing and configuration decisions. By accounting for maintenance-related uncertainty at the design stage, the proposed approach provides a more realistic representation of industrial practice and demonstrates that explicitly modeling maintenance time variability can alter both the optimal number of blocks and workstation configurations. Cycle time is selected as the objective function since maintenance-induced variability directly manifests itself through increased cycle times and throughput loss.

The main contributions of this thesis can be summarized as follows. First, a stochastic formulation of the Type-II Transfer Line Balancing Problem is developed in which workstation maintenance times are explicitly modeled as

random variables rather than deterministic parameters. Second, two exact optimization frameworks—a stochastic programming model and a constraint programming model—are proposed to minimize expected cycle time under maintenance uncertainty. Third, a specialized Branch and Bound algorithm incorporating problem-specific bounding mechanisms and dominance rules is developed. Fourth, to address large-sized instances, matheuristic solution approaches that combine mathematical modeling with structured search and improvement procedures are introduced.

The remainder of this thesis is organized as follows. Chapter 2 reviews the relevant literature on stochastic assembly line balancing, transfer line balancing, and reconfigurable transfer line design. Chapter 3 formally defines the stochastic transfer line balancing problem and presents the stochastic programming and constraint programming models. Chapter 4 introduces the Branch and Bound solution approach. Chapter 5 presents heuristic solution methods along with improvement strategies. Chapter 6 reports computational experiments and discusses the results. Finally, Chapter 7 concludes the study.

CHAPTER 2

LITERATURE REVIEW

The classical assembly line balancing problem (ALBP) focuses on assigning operations to workstations while satisfying precedence constraints and a predefined cycle time. The simplest and most widely studied version is the Simple Assembly Line Balancing Problem (SALBP), originally introduced by Salveson (1955). Comprehensive surveys and classifications of SALBP variants can be found in Battaia and Dolgui (2013).

The transfer line balancing problem (TLBP) addresses the assignment of operations to the blocks and the assignment of the blocks to the workstations while meeting more complex precedence relations. In the absence of additional constraints—specifically, when there is only one operation per block and the number of blocks is unlimited—the TLBP reduces to the SALBP.

Once a transfer line is established, the critical challenge involves allocating operations to blocks and blocks to workstations to maximize the output rate while minimizing cycle time. Maximizing total yield and profit is essential to justify the substantial costs associated with line installation. Moreover, the high capital investment required to establish a transfer line necessitates robust systems capable of enduring unexpected disruptions, such as workstation breakdowns, product modifications, or fluctuating demand. In response to these challenges, RMS can be constructed by incorporating modular hardware and software process modules that allow for quick and reliable rearrangement or replacement. This system architecture offers customized flexibility and an open-ended design, enabling improvements, upgrades, and reconfiguration instead of complete replacement. The primary objective of RMS is to provide the exact functionality and capacity required, precisely when it is needed. Consequently, the main goals include

reducing the lead time for launching new or reconfiguring existing systems, facilitating rapid manufacturing modifications, and ensuring the swift integration of new technologies or functions.

In this study, we will consider designing a transfer line by considering stochasticity on maintenance time of workstations. To position the present study within the existing body of knowledge and to clearly identify the research gap it addresses, this chapter reviews the literature in three streams: (i) stochastic assembly line balancing problems, (ii) deterministic transfer line balancing problems, and (iii) reconfigurable transfer line balancing problems. This structured review highlights that, despite significant advances in modeling flexibility and uncertainty, stochastic maintenance times have not yet been explicitly incorporated into the design of transfer lines.

2.1 Literature on Stochastic Assembly Line Problem

The first study on the assembly line balance problem was published by Salveson (1955). Since then, the linked issues have grown in popularity. As a result of the manufacturing line industry's continuous technological advancements, these related issues are still being researched in the literature.

It should be noted that most of these line balancing studies presuppose complete knowledge of all data. However, to meet the production targets today, the variability regarding the corresponding parameters should be taken into consideration. For example, because of the high variability in operation completion times, each workstation's cycle time is unbalanced, which causes a significant delay in the lead time for product delivery and lowers line productivity. Therefore, it is essential to develop an efficient methodology to manage variability in line to increase line efficiency and lower penalty costs associated with delivery delays. Consequently, incorporating protective mechanisms against uncertainty—often

through stochastic programming and robust optimization—has become a vital area of research.

Early approaches focused on heuristic methods to manage this uncertainty. Shin and Min (1991) proposed a heuristic approach for solving stochastic SALB problems to balance the expected total cost and the highest work completion for the given cycle time throughout a feasible range. Lyu (1997) devised an optimization approach to identify the ideal parameters of a simulation model for addressing stochastic SALB problems. Sarin et al. (1999) proposed a heuristic enumeration technique to reduce the overall cost (labor cost and incompleteness cost) for a stochastic assembly line problem. Their method generates initial results using a dynamic programming procedure, which are then enhanced via a branch and bound enumeration technique.

Subsequent studies proposed increasingly sophisticated solution approaches. Liu et al. (2004) proposed a bidirectional heuristic algorithm to solve the stochastic assembly line balancing problem by assigning operations to workstations in the first stage and smoothing the workload by swapping operations among workstations in the second stage. Cakir et al. (2011) proposed a multi-objective optimization method for a stochastic assembly line balancing problem with a parallel workstation with two objectives: Smoothness index and design cost minimization. A heuristic solution algorithm is developed based on simulated annealing to determine Pareto-optimal solutions for the problem and to avoid revisiting recently searched solutions a tabu list is used, efficiency is improved by repair algorithms and a diversification technique is used to boost the likelihood of finding more Pareto-optimal solutions.

Roy (2011) proposed a mathematical programming approach to balance assembly lines with the dual objectives of minimizing balancing loss and system loss. Balancing loss typically results from the uneven allocation of work, whereas system loss arises from variable operation times—specifically, scenarios where one

workstation has high idle time while another has none, creating flow disruptions. A stochastic framework was developed to model these scenarios effectively.

More recent studies have emphasized robustness and risk mitigation. Hazır and Dolgui (2013) studied line balancing under uncertainty and robust optimization was applied to create designs that protect against disruptions by considering interval uncertainty for operation times. Krishnan and Almaktoom (2016) developed a heuristic method to balance the assembly line in the case of high variability in operations. A risk index approach was created to measure risk and a mathematical model has been developed using the estimated risk to minimize the largest difference in risk between workstations. The solution heuristic of the model evaluates risk through a defined risk index and redistributes workloads across workstations in order to mitigate the stochastic variability and associated risk inherent to each workstation. Lolli et al. (2017) made an effort to maintain the algorithmic structure of a stochastic assembly line balancing by evaluating the changes imposed on the solution by the learning effect over time. They utilized the Kottas-Lau heuristic, which categorizes operations to recalculate time intervals based on operator learning. Finally, Li et al. (2022) proposed a two-stage stochastic programming model for the intertwined problem of assembly line balancing and a capacitated lot-sizing problem with high variability in demand to balance the trade-off between the expected total cost and a downside risk measure. Decomposition of the problem into a series of several smaller problems is beneficial as the solution approach.

As summarized in Table 1, various heuristic, mathematical programming, and hybrid approaches have been proposed to challenge stochasticity and to optimize multiple performance criteria.

Table 1: Literature on Stochastic Assembly Line Balancing Problem

| Author(s), Year | Problem | Solution Approach | Objective Function |
|----------------------------|-----------------|--------------------------|---|
| Shin and Min (1991) | Stochastic SALB | Heuristic approach | Balance total cost and work completion |

| Author(s), Year | Problem | Solution Approach | Objective Function |
|-----------------------------|--|--|---|
| Lyu (1997) | Stochastic SALB | Optimization algorithm | Identify ideal parameters to handle task time variability |
| Sarin et al. (1999) | Stochastic SALB | Heuristic enumeration + dynamic programming + branch & bound | Minimize labor cost and incompleteness cost |
| Liu et al. (2004) | Stochastic SALB | Heuristic (task assignment + workload smoothing) | Balance workload across workstations |
| Cakir et al. (2011) | Stochastic SALB with parallel workstations | Simulated annealing, tabu search, repair & diversification | Minimize smoothness index and design cost (multi-objective) |
| Roy (2011) | Stochastic SALB with variable operation times | Mathematical programming | Minimize balancing and system loss (idle time effects) |
| Hazır & Dolgui (2013) | Stochastic SALB | Mathematical programming | Create robust designs against disruptions |
| Krishnan & Almaktoom (2016) | Stochastic SALB with high operation time variability | Heuristic with risk index analysis | Minimize difference in risk across workstations |
| Lolli et al. (2017) | Stochastic SALB with learning effect | Kottas-Lau heuristic (task categorization, time recalculation) | Adapt solutions considering operator learning over time |
| Li et al. (2022) | Stochastic SALB with demand variability | Two-stage stochastic programming + decomposition | Balance expected total cost and downside risk measure |

2.2 Literature on Transfer Line Balancing Problem

Dolgui et al. (1999) pioneered the study of the TLBP. Their work focused on minimizing the cost of opening blocks and workstations while satisfying operational and technological constraints. To solve this, they suggested Mixed Integer Programming (MIP) and a shortest path approach, effectively modeling the initial problem as a constrained shortest path problem to obtain exact solutions.

Dolgui et al. (2005a) suggested two heuristic algorithms based on the COMSOAL technique that is introduced by Arcus (1966). The first algorithm is named as Recursive Assignment of Predecessors (RAP) and the second algorithm is named as First Satisfy Inclusion Constraints (FSIC). In both methods, operations are assigned to the blocks in workstations randomly and best assignment is kept. According to the RAP algorithm, a feasible operation is chosen randomly, then, its

predecessors and inclusion constraint pairs are assigned. If this operation cannot be assigned to a block, a new block is formed and if cannot be assigned to a workstation, a new workstation is created. The algorithm controls the workstation exclusion constraints after the workstation is filled. Therefore, some undesirable solutions are generated. The FSIC algorithm focuses on eliminating these infeasible solutions by processing an operation's inclusion constraint pairs and unassigned predecessors. The results of their experiment showed that FSIC algorithm is more effective than the RAP algorithm.

Dolgui et al. (2005b) developed a MIP model and propose mechanisms to reduce its size. They also introduced decomposition heuristic procedure for large-sized problems. The mechanisms for the reduction of the model size are similar to the SALBP study of Patterson and Albracht (1975) but more complex due to the inclusion and exclusion constraints. In the decomposition heuristic procedure, the initial set of operations are partitioned into smaller sets and then these small-sized sets are separately solved by using MIP model. Guschinskaya et al. (2005) further advanced this by providing a hybrid method based on decomposition. In their approach, FSIC is used to find an initial feasible solution and assignment path; the decomposition is then structured around this path. Each resulting subproblem is solved using the shortest path approach, and the global solution is updated based on the subproblem results.

Dolgui et al. (2006) conducted a comprehensive comparison of these solution methods, including MIP, shortest path, FSIC, decomposition methods, and hybrid approaches. Guschinskaya and Dolgui (2009) extended this comparison by investigating deterministic decomposition based on precedence graphs and a heuristic multi-start decomposition (an improvement of the hybrid method that applies FSIC multiple times). Both studies concluded that for small-sized instances, the shortest path method yields better solutions in terms of both computational time and quality. However, for medium and large-sized instances, the hybrid and FSIC methods outperform their competitors.

Regarding exact methods, Dolgui (2009) developed a lower bound (LB) based set partitioning problem and applied branch and bound (B&B) algorithm. B&B algorithm is proposed whose efficiency is enhanced by LB and dominance rule. It is close to our B&B method. In fact, Klein and Scholl (1996) provided the main concepts of B&B for SALBP. Determination of LB and upper bound (UB) is introduced and possible improvements of these bounds remarked. B&B is an implicit enumeration method, explained in detail by examining fathoming rules, backtracking, and efficient selections. The B&B method developed in this thesis draws upon the concepts presented in this paper.

In a review of trends and challenges, Dolgui (2010) distinguished between Dedicated Transfer Lines (DTL) for fixed, known demand and Flexible Transfer Lines (FTL) for diverse, unknown demand. However, given that even advanced systems can become obsolete due to large market variations and demand uncertainty, Reconfigurable Transfer Lines (RTL) have been proposed. RTLs are designed to adapt to changes in market variations, with some authors suggesting they offer greater flexibility than FTLs. Consequently, the reconfiguration and rebalancing of transfer lines are foreseen as attractive research topics.

Battaia et al. (2016) considered a joint formulation of process planning and transfer line design for mixed model production. A mathematical model is proposed which is tested on real-world industrial problem. Finally, for a detailed and contemporary analysis of the TLBP landscape, the review by Beldar et al. (2025) serves as a key reference.

Table 2 provides an overview of the major contributions in the literature on the TLBP, outlining the addressed problem variants, the methodological approaches employed, and the primary objective functions pursued in each study.

Table 2: Literature on Transfer Line Balancing Problem

| Author(s), Year | Problem | Solution Approach | Objective Function |
|------------------------|--------------------|-----------------------------|---------------------------|
| Dolgui et al. (1999) | TLBP (First study) | MIP, Shortest path approach | Cost minimization |

| Author(s), Year | Problem | Solution Approach | Objective Function |
|--------------------------------|---|---|---|
| Dolgui et al. (2005a) | TLBP | Heuristic approach | Cost minimization |
| Dolgui et al. (2005b) | TLBP | MIP with size reduction, decomposition heuristic | Cost minimization |
| Guschinskaya et al. (2005) | Decomposition of TLBP | Hybrid method: FSIC + shortest path decomposition | Cost minimization |
| Dolgui et al. (2006) | Comparison of solution methods for TLBP | MIP, shortest path, FSIC, decomposition, hybrid | Solution quality and computational efficiency |
| Guschinskaya and Dolgui (2009) | Comparison of solution methods for TLBP | FSIC, hybrid, decomposition | Solution quality and computational efficiency |
| Dolgui (2009) | TLBP as set partitioning problem | B&B algorithm (LB + dominance rules) | Cost minimization |
| Dolgui (2010) | Trends and challenges in TLBP | Conceptual analysis | Flexibility, adaptability |
| Battaia et al. (2016) | TLBP with different parts | Mathematical programming | Cost minimization |

2.3 Literature on Reconfigurable Transfer Line Balancing Problem

Reconfigurable manufacturing systems (RMS) are designed to accommodate variations in market demand and unexpected machine breakdowns through simple, rapid adjustments. Ideally, RMS offers the precise functionality and capacity required at any given time and can be cost-effectively modified as needed. To achieve this, the system must possess key characteristics: modularity, integrability, customizability, convertibility, and diagnosability. The drive for mass customization and the prevalence of shorter production runs serve as the primary forces behind the implementation of these lines.

Koren et al. (1999) introduced the novel concept of the RMS in the late 1990s, and it began to gain significant attention in the literature by the early 2000s. Mehrabi et al. (2000) reviewed existing manufacturing methods and established RMS as a new paradigm enabling quick adjustments to production capacity and functionality in response to changing market conditions. Their study investigated various manufacturing techniques, highlighting their achievements and limitations, and accurately predicted the rising importance of reconfigurable systems.

In the domain of optimization, Bratcu et al. (2005) investigated the issue of minimizing the cost of reconfigurable transfer lines which are used to produce a family of products. A linear programming approach is presented, and pre-processed methods are used for model reduction along with enhancing the bounds. Essafi et al. (2012) addressed the balancing reconfigurable transfer lines with sequence-dependent setup times and parallel machines and a heuristic solution based on the greedy randomized adaptive search procedure (GRASP). Self-adjusted parameters have been added to the algorithm and it is enforced to be reactive, and it is also improved with the path relinking procedure.

Borisovsky et al. (2013) proposed reducing the reconfigurable TLBP to a set partitioning problem by generating every workstation, and for each one, the best schedule in which to do the necessary operations are enumerated by benefiting a dynamic programming algorithm. A series of well-known techniques like constraint generation and branch and cut algorithms are suggested to solve the given set partitioning type problem. Their parallel Branch and Cut algorithms demonstrated strong computational efficiency even for medium-to-large-sized instances.

More recent studies have incorporated multiple objectives and uncertainty. Delorme et al. (2016) studied the balancing of reconfigurable transfer lines by considering two main objectives: cycle time and total cost. They provided a GRASP-based heuristic, utilizing Mixed Integer Programming (MIP) to solve subproblems and local search algorithms to enhance performance. The approach generates a Pareto front of non-dominated, efficient solutions.

Liu et al. (2019) addressed balancing with demand uncertainty through a two-stage optimization model. The first stage focuses on designing an initial reconfigurable line under unknown demand—determining the number of workstations, parallel machines, and configurations before sequencing operations. The second stage aims to reconfigure the initial line (by adding or removing machines) to minimize

reconfiguration and lost sales costs, transforming the interval demand values from the first stage into discrete determined values. A hybrid approach combining GRASP with Variable Neighborhood Search (VNS) was proposed.

Lahrichi et. al. (2020) suggested a new MIP approach based on the formulation for the sequence-dependent setup time assembly line balancing problem introduced by Andres et al. (2008). Using preprocessing procedures, they extracted redundancies to streamline the model. Furthermore, they presented a hybrid "balance first, sequence last" strategy and provided a simple lower bound method for practical application.

Finally, Battaia et al. (2023) addressed the challenging problem of optimizing the configuration of a rotary transfer machine with turrets for machining multiple parts. Optimization involves decisions such as selecting part orientations on the rotary table, assigning operations to machining modules, and determining cutting modes for spindle heads and turrets. The goal is to minimize the overall equipment cost. Given the combinatorial nature of the problem, they introduced a heuristic framework to assist decision-makers in managing multi-part batches efficiently. The methodology was validated on real-world industrial cases, demonstrating its effectiveness in minimizing machine design costs.

Table 3 presents a summary of the key studies addressing the Reconfigurable Transfer Line Balancing Problem (RTLBP), highlighting each work’s focus, the solution approach adopted, and the corresponding objective functions.

Table 3: Literature on Reconfigurable Transfer Line Balancing Problem

| Author(s), Year | Problem | Solution Approach | Objective Function |
|------------------------|---|--|---------------------------------------|
| Koren et al. (1999) | Introduction of RMS | Conceptual framework | Quick reconfiguration, modularity |
| Mehrabi et al. (2000) | Review of Reconfigurable Manufacturing System | Comparative analysis of methods | Flexibility, adaptability to changes |
| Bratcu et al. (2005) | Reconfigurable TLBP | LP, model reduction, bound enhancement | Minimize total cost |
| Essafi et al. (2012) | Reconfigurable TLBP with sequence-dependent setup | GRASP heuristic with self-adjusted parameters, | Efficient balancing under setup times |

| Author(s), Year | Problem | Solution Approach | Objective Function |
|--------------------------|--|--|---|
| | times and parallel machines | path relinking | |
| Borisovsky et al. (2013) | Reconfigurable TLBP as set partitioning | Dynamic programming, constraint generation, branch and cut | Minimize total cost |
| Delorme et al. (2016) | Reconfigurable TLBP | GRASP heuristic + MIP + local search | Pareto optimization: minimize cycle time & cost |
| Liu et al. (2019) | Reconfigurable TLBP with demand uncertainty | Two-stage optimization model + GRASP with VNS | Minimize reconfiguration & order loss costs |
| Lahrichi et al. (2020) | Reconfigurable TLBP | MIP + hybrid approach (balance and sequence) | Minimize cycle time & setup-related costs |
| Battaia et al. (2023) | Optimizing configuration of rotary transfer machines | Heuristic approach | Minimize overall equipment cost |

2.4 Contribution of the Thesis

The Transfer Line Balancing Problem (TLBP) has long been a central research topic in the design and operation of automated manufacturing systems. Early studies primarily addressed deterministic and static models where the key objective was to allocate operations to blocks and workstations under fixed cycle times and precedence relations, focusing on minimizing line installation costs or cycle time. These works established the fundamental structure of transfer line balancing and highlighted its combinatorial complexity. However, as manufacturing environments have evolved, characterized by shorter product life cycles, demand variability, disruptions and increasing customization requirements, the assumptions of deterministic and static models have become inadequate. In response, a stream of research has emerged focusing on reconfigurable transfer lines and stochastic extensions of line balancing problems. These studies were instrumental in moving the literature beyond purely deterministic formulations.

A more systematic treatment of uncertainty in reconfigurable transfer lines has been introduced recently. While these works have advanced the literature considerably, several limitations remain. First, most studies addressing uncertainty in transfer lines focus almost exclusively on demand variability or on deterministic but complex operational constraints such as sequence-dependent setups. Other

critical sources of variability, such as workstation maintenance times, have received little to no attention, despite their significant influence on effective cycle time and throughput in real-world manufacturing. Variability in these parameters directly affects line reliability, availability, and the need for corrective reconfigurations.

The present study makes novel contribution by explicitly incorporating of stochastic workstation maintenance times into the design stage of the TLBP. To the best of our knowledge, no previous study has simultaneously addressed this form of stochasticity within the reconfigurable transfer line problem while aiming to minimize cycle time. By embedding these uncertainties into the configuration stage, the thesis proposes a proactive framework that anticipates variability before it leads to costly reconfigurations. This shifts the focus from purely reactive adaptation toward robust-reconfigurable design, where the line is inherently structured to absorb and mitigate common sources of variability.

The thesis develops an integrated solution framework that leverages multiple optimization paradigms. Specifically, stochastic and constraint programming are employed to model uncertainty rigorously, while both exact and heuristic solution strategies are proposed. An exact B&B algorithm is developed to solve smaller instances to optimality, providing a benchmark for evaluating heuristic performance. In addition, heuristic methods, including B&B- and Model-based heuristics, are designed to efficiently tackle larger and more complex problem instances. This hybrid framework ensures theoretical soundness while maintaining computational scalability and it offers a more comprehensive toolkit than the predominantly heuristic-focused approaches seen in prior studies.

By addressing this critical gap and offering both methodological and conceptual innovations, the thesis advances academic understanding of the Type-II Stochastic TLBP. Furthermore, it provides practical insights for the design of resilient, adaptable, and cost-effective transfer lines in today's volatile industrial environment.

CHAPTER 3

STOCHASTIC TRANSFER LINE BALANCING PROBLEM

In this section, we define our problem, review the fundamentals of the stochastic programming approach, and present the stochastic and constraint programming models. We also provide the mathematical models used to evaluate the performance of the stochastic programming model.

3.1 Problem Statement

TLBP involves assigning a given set of operations to a predefined number of workstations with the aim of optimizing overall production performance. The primary objective of this study is to minimize the cycle time while ensuring that the workloads are reasonably balanced among the workstations. A well-balanced line reduces idle times, improves throughput, and mitigates the risk of production bottlenecks, thereby increasing the overall efficiency of the system.

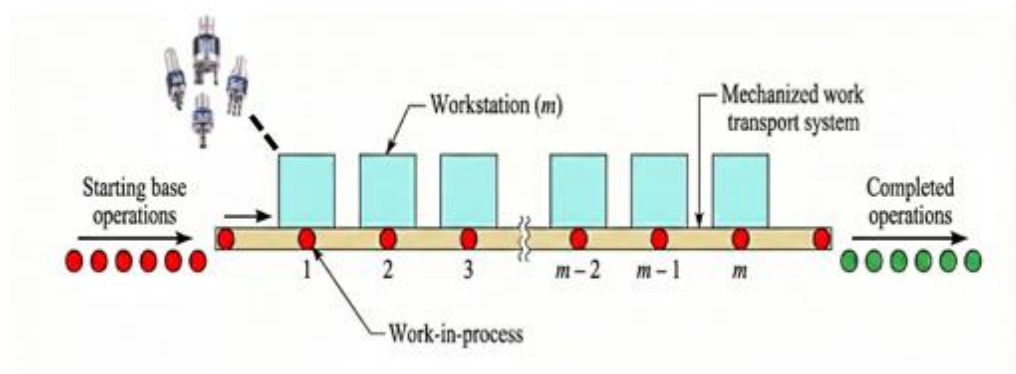


Figure 2: Transfer Line

A total of n operations must be assigned to these workstations. Each operation i requires a deterministic processing time t_i . The assignment process is hierarchical:

operations are first grouped into blocks, and then blocks are assigned to workstations. All operations within the same block are performed in parallel. It is assumed that operations are indivisible; hence, each operation must be assigned to exactly one block.

Due to technological limitations, each block can accommodate at most n_o operations, and each workstation can process at most m_o blocks. These restrictions reflect practical constraints such as the limited number of spindle heads and the physical capacity of the workstations.

The precedence relations among operations are defined as follows: If operation j is a predecessor of operation i , then operation i cannot start until operation j has been completed. Consequently, operation i must be assigned to a block that is processed strictly later than the block containing operation j . Let IP_i denote the set of immediate predecessors of operation i , and let P_i denote the set of all (direct and indirect) predecessors of operation i . Then, $j \in IP_i$ and P_i .

In addition to precedence constraints, three types of technological constraints are considered. First, block exclusion constraints specify pairs of operations that cannot be assigned to the same block, typically due to incompatible setup requirements. Let EB denote the set of such operation pairs. Second, workstation exclusion constraints indicate pairs of operations that cannot be processed at the same workstation, for instance, due to incompatible resource requirements; this set is denoted by ES . Third, workstation inclusion constraints require certain pairs of operations to be processed at the same workstation due to technological or operational necessities. The set of such pairs is denoted by IS .

Furthermore, we assume that each workstation may require maintenance at the beginning of its processing period. These maintenance activities may include cleaning, lubrication, minor adjustments, or the replacement of small components. The duration of such activities is uncertain and may vary from a few seconds to several minutes. For instance, a quick lubrication may take only a few seconds,

while a minor part replacement may take minutes and a more detailed adjustment may be required longer times. This uncertainty is modeled using a discrete probability distribution with known parameters.

The objective of this study is to determine both the assignment of operations to blocks and the assignment of blocks to workstations so as to minimize the expected cycle time of the transfer line, explicitly accounting for the stochastic nature of maintenance durations of workstations.

3.2 Brief Background on Stochastic Programming

Stochastic programming is a mathematical optimization framework designed to address decision-making problems that involve uncertain parameters whose probability distributions are assumed to be known. This approach is particularly suitable for modeling real-life systems in which uncertainty plays a significant role in system performance.

In a two-stage stochastic programming framework, decision variables are classified into first-stage and second-stage decisions. First-stage decisions are made before the realization of uncertainty, whereas second-stage decisions, also referred to as recourse decisions, are determined after the uncertain parameters become known.

A scenario represents a specific realization of the uncertain parameters. Let S denote the set of all possible scenarios, indexed by $s \in S$. The first-stage decision variables are denoted by $x \in X$, while the second-stage decision variables corresponding to scenario s are denoted by $y_s \in Y$ for all scenarios.

The objective function of a two-stage stochastic program may consist of a deterministic first-stage component and an expected second-stage component, or it may solely depend on the expected value of the second-stage outcomes, depending on the structure of the problem.

The extensive form of a two-stage stochastic minimization problem can be expressed as follows:

$$\text{Min } Z = cx + \sum_{s \in S} p_s q_s y_s$$

subject to

$$Ax = b$$

$$T_s x_s + W_s y_s = h_s \quad \forall s \in S$$

$$x \in X$$

$$y_s \in Y, \quad \forall s \in S$$

where p_s is the realization probability of scenario s . c, A, B, q_s, T_s, W_s and h_s are real valued vectors and matrices.

The computational complexity of stochastic programs strongly depends on the number of scenarios considered. When the number of scenarios is small, such problems can often be solved using standard linear or integer programming techniques. However, as the number of scenarios increases, the resulting models may become computationally intractable, requiring the use of decomposition techniques or scenario reduction methods.

3.3 Stochastic Transfer Line Balancing Problem

Our stochastic transfer line balancing problem considers the uncertainty associated with maintenance times at any given workstation. These maintenance times can represent various factors, such as the actual maintenance duration or the recovery time following a breakdown. Maintenance times are assumed to be equal for every workstation implying workstations are identical.

We consider two cases:

Case 1: two types of breakdowns: no breakdown and breakdown

Case 2: three types of breakdowns: no breakdown, minor breakdown, and major breakdown

Additionally, we consider two scenarios for the breakdown probabilities:

1. Identical probabilities: Workstations share the same probability for breakdowns.
2. Non-identical probabilities: Workstations have heterogeneous probabilities.

CASE 1

For Case 1, we assume that there are two possible types of maintenance time:

Type 1; Maintenance time is zero, including no breakdown.

Type 2; Maintenance time is u_k for workstation k , indicating a breakdown.

Let π_{kh} be the probability that type h maintenance (based in its breakdown type) is required for workstation k . This satisfies the condition: $\sum_{h=1}^2 \pi_{kh} = 1$ for all k . This implies that exactly one type of breakdown occurs at each workstation. If a Type 1 breakdown occurs at all workstations, then no maintenance is required.

We assume that breakdowns are independent. Therefore, the probability that breakdown type h_1 occurs at workstation 1 and type h_2 occurs at workstation 2 and so on is $\pi_{1h_1} \cdot \pi_{2h_2} \cdots \pi_{mh_m} = \prod_{k=1}^m \pi_{kh_k}$.

The probability that type 1 breakdown occurs at all workstations, i.e., $h_k = 1$ for all k , is $\pi_{11} \cdot \pi_{21} \cdots \pi_{m1} = \prod_{k=1}^m \pi_{k1}$.

The number of possible scenarios, S , is 2^m since there are m workstations and two breakdown types for each workstation. For example, when there are three workstations, the number of scenarios is $2^3 = 8$, and we let p_s be the probability of scenario s , $s = 1, \dots, S$ ($S = 8$ in this case).

Table 4 presents Case 1 scenarios and related probabilities.

Table 4: Case 1 scenarios and related probabilities

| Scenario s | Workstation / Breakdown Type | | | Probability p_s |
|-----------------|------------------------------|---|---|--|
| | 1 | 2 | 3 | |
| 1 | 1 | 1 | 1 | $\pi_{11} \cdot \pi_{21} \cdot \pi_{31}$ |
| 2 | 1 | 1 | 2 | $\pi_{11} \cdot \pi_{21} \cdot \pi_{32}$ |
| 3 | 1 | 2 | 1 | $\pi_{11} \cdot \pi_{22} \cdot \pi_{31}$ |
| 4 | 1 | 2 | 2 | $\pi_{11} \cdot \pi_{22} \cdot \pi_{32}$ |
| 5 | 2 | 1 | 1 | $\pi_{12} \cdot \pi_{21} \cdot \pi_{31}$ |
| 6 | 2 | 1 | 2 | $\pi_{12} \cdot \pi_{21} \cdot \pi_{32}$ |
| 7 | 2 | 2 | 1 | $\pi_{12} \cdot \pi_{22} \cdot \pi_{31}$ |
| 8 | 2 | 2 | 2 | $\pi_{12} \cdot \pi_{22} \cdot \pi_{32}$ |

CASE 2

For Case 2, we assume that there are three possible types of maintenance time:

Type 1; Maintenance time is zero, indicating no breakdown.

Type 2; Maintenance time is u_{1k} for workstation k , indicating a minor breakdown.

Type 3; Maintenance time is u_{2k} where $u_{2k} > u_{1k}$ for workstation k , indicating a major breakdown.

Let π_{kh} denote the probability that maintenance type h maintenance (based in its breakdown type) is required for workstation k . This satisfies the following condition:

$\sum_{h=1}^3 \pi_{kh} = 1$ for all k which implies that exactly one type of breakdown occurs at each workstation.

We assume that breakdowns are independent. Therefore, the probability that breakdown type h_1 occurs at workstation 1, type h_2 occurs at workstation 2, type h_3 occurs at workstation 3, and so on, is given by: $\pi_{1h_1} \cdot \pi_{2h_2} \dots \pi_{mh_m} = \prod_{k=1}^m \pi_{kh_k}$. Accordingly, the probability that a Type 3 breakdown occurs at all workstations, i.e., $h_k = 3$ for all k , is $\pi_{13} \cdot \pi_{23} \dots \pi_{m3} = \prod_{k=1}^m \pi_{k3}$.

The total number of scenarios, S , is 3^m since three breakdown types for each workstation. For example, when there are three workstations, the total number of scenarios is $3^3 = 27$. Let p_s denote the probability of scenario s .

Table 5 presents Case 2 scenarios and related probabilities.

Table 5: Case 2 scenarios and related probabilities

| Scenario s | Workstation / Breakdown Type | | | Probability p_s |
|-----------------|------------------------------|---|---|--|
| | 1 | 2 | 3 | |
| 1 | 1 | 1 | 1 | $\pi_{11} \cdot \pi_{21} \cdot \pi_{31}$ |
| 2 | 1 | 1 | 2 | $\pi_{11} \cdot \pi_{21} \cdot \pi_{32}$ |
| 3 | 1 | 1 | 3 | $\pi_{11} \cdot \pi_{21} \cdot \pi_{33}$ |
| 4 | 1 | 2 | 1 | $\pi_{11} \cdot \pi_{22} \cdot \pi_{31}$ |
| 5 | 1 | 2 | 2 | $\pi_{11} \cdot \pi_{22} \cdot \pi_{32}$ |
| 6 | 1 | 2 | 3 | $\pi_{11} \cdot \pi_{22} \cdot \pi_{33}$ |
| 7 | 1 | 3 | 1 | $\pi_{11} \cdot \pi_{23} \cdot \pi_{31}$ |
| 8 | 1 | 3 | 2 | $\pi_{11} \cdot \pi_{23} \cdot \pi_{32}$ |
| 9 | 1 | 3 | 3 | $\pi_{11} \cdot \pi_{23} \cdot \pi_{33}$ |
| 10 | 2 | 1 | 1 | $\pi_{12} \cdot \pi_{21} \cdot \pi_{31}$ |
| 11 | 2 | 1 | 2 | $\pi_{12} \cdot \pi_{21} \cdot \pi_{32}$ |
| 12 | 2 | 1 | 3 | $\pi_{12} \cdot \pi_{21} \cdot \pi_{33}$ |
| 13 | 2 | 2 | 1 | $\pi_{12} \cdot \pi_{22} \cdot \pi_{31}$ |
| 14 | 2 | 2 | 2 | $\pi_{12} \cdot \pi_{22} \cdot \pi_{32}$ |
| 15 | 2 | 2 | 3 | $\pi_{12} \cdot \pi_{22} \cdot \pi_{33}$ |
| 16 | 2 | 3 | 1 | $\pi_{12} \cdot \pi_{23} \cdot \pi_{31}$ |
| 17 | 2 | 3 | 2 | $\pi_{12} \cdot \pi_{23} \cdot \pi_{32}$ |
| 18 | 2 | 3 | 3 | $\pi_{12} \cdot \pi_{23} \cdot \pi_{33}$ |
| 19 | 3 | 1 | 1 | $\pi_{13} \cdot \pi_{21} \cdot \pi_{31}$ |
| 20 | 3 | 1 | 2 | $\pi_{13} \cdot \pi_{21} \cdot \pi_{32}$ |
| 21 | 3 | 1 | 3 | $\pi_{13} \cdot \pi_{21} \cdot \pi_{33}$ |
| 22 | 3 | 2 | 1 | $\pi_{13} \cdot \pi_{22} \cdot \pi_{31}$ |
| 23 | 3 | 2 | 2 | $\pi_{13} \cdot \pi_{22} \cdot \pi_{32}$ |
| 24 | 3 | 2 | 3 | $\pi_{13} \cdot \pi_{22} \cdot \pi_{33}$ |
| 25 | 3 | 3 | 1 | $\pi_{13} \cdot \pi_{23} \cdot \pi_{31}$ |
| 26 | 3 | 3 | 2 | $\pi_{13} \cdot \pi_{23} \cdot \pi_{32}$ |
| 27 | 3 | 3 | 3 | $\pi_{13} \cdot \pi_{23} \cdot \pi_{33}$ |

3.4 Finding the Earliest and Latest Workstation Values:

In this section, we derive expressions for the earliest and latest workstations to which operations can be assigned, based on precedence constraints and workstation inclusion constraints.

To determine the earliest and latest workstation for an operation, it is not sufficient to consider only its own precedence relations. One must also account for the precedence relations of other operations that are required to be assigned to the same workstation. Therefore, we partition the set of operations into subsets according to the inclusion constraints.

Let IS_r denote the r^{th} subset of operations that must be assigned to the same workstation, where $r = 1, \dots, n$.

Example 1 illustrates the IS_r sets:

Example 1: $n = 15$ and $IS = \{(10,11), (11,12), (13,14)\}$. Then the inclusion subsets are $IS_1 = \{1\}, IS_2 = \{2\} \dots IS_9 = \{9\}, IS_{10} = \{10, 11, 12\}, IS_{11} = \{13, 14\}, IS_{12} = \{15\}$.

We distinguish two cases:

Case 1: $|IS_r| = 1$

Case 2: $|IS_r| \geq 2$

Theorem 1a. There is no feasible solution in which operation i in set r such that

$|IS_r| = 1$ is assigned to workstations $1, 2, \dots, E_i - 1$ where $E_i = \left\lceil \frac{\lceil \frac{|P_i|}{n_0} \rceil + 1}{m_0} \right\rceil$

Proof. $\left\lceil \frac{|P_i|}{n_0} \right\rceil$ is a lower bound on the number of blocks that should be opened before operation i is assigned, as there are $|P_i|$ predecessors of operation i , and each block can accommodate at most n_0 operations. Together with operation i there will be at

least $\left\lceil \frac{|P_i|}{n_0} \right\rceil + 1$ blocks, hence at least $\left\lceil \frac{\lceil \frac{|P_i|}{n_0} \rceil + 1}{m_0} \right\rceil$ workstations, as each workstation can

handle m_0 blocks.

Theorem 1b. There is no feasible solution in which any operation in set r such that $|IS_r| \geq 1$ is assigned to workstations $1, 2, \dots, E_r - 1$ where $E_r =$

$$\max \left\{ \left\lceil \frac{\left\lceil \frac{|U_{i \in IS_r} P_i|}{n_0} \right\rceil + 1}{m_0} \right\rceil, \left\lceil \frac{|U_{i \in IS_r} P_i U_{IS_r}|}{n_0 * m_0} \right\rceil \right\}.$$

Proof. $\left\lceil \frac{|U_{i \in IS_r} P_i|}{n_0} \right\rceil$ is a lower bound on the number of blocks that should be opened before any operation in IS_r . This is due to the fact that there are $|U_{i \in IS_r} P_i|$ predecessors over all operations in IS_r and each block can accommodate at most n_0 operations. Together with operation i there will be at least $\left\lceil \frac{|U_{i \in IS_r} P_i|}{n_0} \right\rceil + 1$ blocks,

hence at least $\left\lceil \frac{\left\lceil \frac{|U_{i \in IS_r} P_i|}{n_0} \right\rceil + 1}{m_0} \right\rceil$ workstations, as each workstation can handle m_0

blocks.

Moreover, to assign an operation in IS_r with other operations in IS_r , we should assign $|U_{i \in IS_r} P_i U_{IS_r}|$ operations. This follows, $\left\lceil \frac{|U_{i \in IS_r} P_i U_{IS_r}|}{n_0 * m_0} \right\rceil$ is the earliest workstation that any operation in IS_r can be assigned, as each workstation accommodates at most m_0 blocks and each block contains at most n_0 operations.

Hence, $\max \left\{ \left\lceil \frac{\left\lceil \frac{|U_{i \in IS_r} P_i|}{n_0} \right\rceil + 1}{m_0} \right\rceil, \left\lceil \frac{|U_{i \in IS_r} P_i U_{IS_r}|}{n_0 * m_0} \right\rceil \right\}$ is the earliest workstation that any

operation in IS_r can be assigned.

Examples 2 and 3 demonstrate the results of Theorems 1a and 1b, respectively:

Example 2: $n = 15, m = 4, n_0 = 2, m_0 = 2$ and $IS = \{(10,11), (11,12), (13,14)\}$. Then, $IS_1 = \{1\}, IS_2 = \{2\} \dots IS_9 = \{9\}, IS_{10} = \{10\}, IS_{11} = \{11, 12, 13, 14\}, IS_{12} = \{15\}$. The set of precedence constraints is $\{(7; 10), (8; 10), (9; 10)\}$.

$$E_{10} = \left\lceil \frac{\left\lceil \frac{|U_{i \in IS_r} P_i|}{n_0} \right\rceil + 1}{m_0} \right\rceil = \left\lceil \frac{2+1}{2} \right\rceil = 2$$

Example 3: $n = 15, m = 4, n_o = 2, m_o = 2$ and $IS = \{(10,11), (11,12), (13,14)\}$. Then, $IS_1 = \{1\}, IS_2 = \{2\} \dots IS_9 = \{9\}, IS_{10} = \{10, 11, 12\}, IS_{11} = \{13, 14\}, IS_{12} = \{15\}$. The set of precedence constraints is $\{(8; 10), (9; 10)\}$.

$$E_{10} = \max \left\{ \left\lceil \frac{\left\lceil \frac{|U_{i \in IS_r} P_i|}{n_0} \right\rceil + 1}{m_0} \right\rceil, \left\lceil \frac{|U_{i \in IS_r} P_i U_{IS_r}|}{n_o * m_o} \right\rceil \right\} = \max \left\{ \left\lceil \frac{1+1}{2} \right\rceil, \left\lceil \frac{5}{4} \right\rceil \right\} = \max\{1, 2\} = 2$$

Theorem 2a. There is no feasible solution in which operation i in set r such that $|IS_r| = 1$ is assigned to workstations $L_i + 1, L_i + 2, \dots, m$ where $L_i = m - \left\lceil \frac{\left\lceil \frac{|S_i|}{n_0} \right\rceil + 1}{m_0} \right\rceil + 1$

Proof. $\left\lceil \frac{|S_i|}{n_0} \right\rceil$ is lower bound on the number of blocks opened after operation i 's block, as there are $|S_i|$ successors of operation i , and each block can accommodate at most n_0 operations. Together with operation i there will be at least $\left\lceil \frac{|S_i|}{n_0} \right\rceil + 1$ blocks, hence at least $\left\lceil \frac{\left\lceil \frac{|S_i|}{n_0} \right\rceil + 1}{m_0} \right\rceil$ workstations since each workstation can handle m_0

blocks. Thus, operation i can be placed at the latest on workstation $m - \left\lceil \frac{\left\lceil \frac{|S_i|}{n_0} \right\rceil + 1}{m_0} \right\rceil + 1$.

Theorem 2b. There is no feasible solution in which any operation in set r such that $|IS_r| \geq 1$ is assigned to workstations $L_i + 1, L_i + 2, \dots, m$ where $L_i = m -$

$$\max \left\{ \left\lceil \frac{\left\lceil \frac{|U_{i \in IS_r} S_i|}{n_0} \right\rceil + 1}{m_0} \right\rceil, \left\lceil \frac{|U_{i \in IS_r} S_i U_{IS_r}|}{n_o * m_o} \right\rceil \right\} + 1.$$

Proof. $\left\lceil \frac{|S_i|}{n_0} \right\rceil$ is lower bound on the number of blocks opened after operation i 's block, as there are $|S_i|$ successors of operation i , and each block can accommodate at most n_0 operations. Together with operation i there will be at least $\left\lceil \frac{|S_i|}{n_0} \right\rceil + 1$ blocks, hence at least $\left\lceil \frac{\left\lceil \frac{|S_i|}{n_0} \right\rceil + 1}{m_0} \right\rceil$ workstations since each workstation can handle m_0 blocks. Thus, operation i can be placed at the latest on workstation $m - \left\lceil \frac{\left\lceil \frac{|S_i|}{n_0} \right\rceil + 1}{m_0} \right\rceil + 1$.

Moreover, to assign an operation in IS_r with other operations in IS_r , we should assign $|\cup_{i \in IS_r} S_i \cup IS_r|$ operations. This follows, $m - \left\lceil \frac{|\cup_{i \in IS_r} S_i \cup IS_r|}{n_0 * m_0} \right\rceil + 1$ is the latest workstation that any operation in IS_r can be assigned, as each workstation accommodates at most m_0 blocks and each block contains at most n_0 operations.

Hence, $m - \max\left\{\left\lceil \frac{\left\lceil \frac{|\cup_{i \in IS_r} S_i|}{n_0} \right\rceil + 1}{m_0} \right\rceil, \left\lceil \frac{|\cup_{i \in IS_r} S_i \cup IS_r|}{n_0 * m_0} \right\rceil\right\} + 1$ is the latest workstation that any operation in IS_r can be assigned.

Examples 4 and 5 demonstrate the results of Theorems 2a and 2b, respectively:

Example 4: $n = 15, m = 4, n_0 = 2, m_0 = 2$ and $IS = \{(10,11), (11,12), (13,14)\}$. Then, $IS_1 = \{1\}, IS_2 = \{2\} \dots IS_9 = \{9\}, IS_{10} = \{10\}, IS_{11} = \{11, 12, 13, 14\}, IS_{12} = \{15\}$. The set of precedence constraints is $\{(10; 11), (10; 12), (10; 13)\}$.

$$E_{10} = m - \left\lceil \frac{\left\lceil \frac{|\cup_{i \in IS_r} S_i|}{n_0} \right\rceil + 1}{m_0} \right\rceil + 1 = 4 - \left\lceil \frac{2+1}{2} \right\rceil + 1 = 3$$

Example 5: $n = 15, m = 4, n_o = 2, m_o = 2$ and $IS = \{(10,11), (11,12), (13,14)\}$. Then, $IS_1 = \{1\}, IS_2 = \{2\} \dots IS_9 = \{9\}, IS_{10} = \{10, 11, 12\}, IS_{11} = \{13, 14\}, IS_{12} = \{15\}$. The set of precedence constraints is $\{(10, 13), (10, 14)\}$.

$$E_{10} = m - \max \left\{ \left\lceil \frac{\left\lceil \frac{|\cup_{i \in IS_r} S_i|}{n_o} \right\rceil + 1}{m_o} \right\rceil, \left\lceil \frac{|\cup_{i \in IS_r} |S_i \cup IS_r||}{n_o * m_o} \right\rceil \right\} + 1 = 4 - \max \left\{ \left\lceil \frac{1+1}{2} \right\rceil, \left\lceil \frac{5}{4} \right\rceil \right\} +$$

$$1 = 3$$

We further strengthen the IS_y and ES by imposing the following conditions:

Inclusion constraints:

If for any $i \in IS_y, j \in IS_z$ and $a \in IS_z, y \neq z$ and operation a is successor of operation j and predecessor of operation i then IS_y and IS_z are merged, i.e., $IS_y = IS_y \cup IS_z$.

Exclusion constraints:

- i. If for any $i \in IS_y$ and $j \in IS_z, y \neq z$ if $|IS_y| + |IS_z| > n_o * m_o$ then operation i and j cannot be assigned to the same workstation, thus the pair (i, j) is added to set ES .
- ii. If $E_i > L_j$ and $E_j > L_i$ then operation i and j cannot be assigned to the same workstation, accordingly, the pair (i, j) is added to set ES .

These strengthening actions are repeated until no such operations exist.

The results derived in Section 3.4 are used to enhance the performance of the models presented in the following section.

3.5 Stochastic Programming Model

The mathematical model is constructed based on this scenario structure. We refer to the associated model as Stochastic Programming (SP) model and utilize the following parameters and decision variables to state the model.

Parameters:

n = number of operations

m = number of workstations

S = number of scenarios

n_o = maximum number of operations per block

m_o = maximum number of blocks per workstation

t_i = processing (operation) time of operation i , $i = 1, \dots, n$

P_i = set of all predecessors of operation i , $i = 1, \dots, n$

ES = set of operation pairs that cannot be assigned to the same workstation

EB = set of operation pairs that cannot be assigned to the same block

IS = set of operation pairs that should be assigned to the same workstation

E_i = the earliest workstation that operation i can be assigned, $i = 1, \dots, n$

L_i = the latest workstation that operation i can be assigned, $i = 1, \dots, n$

u_{ks} = maintenance time of workstation k in scenario s , $s = 1, \dots, S$, $k = 1, \dots, m$

p_s = realization probability of scenario s , $s = 1, \dots, S$

First-stage decision variables:

The first-stage decision variables determine the assignment of operations to blocks and workstations before the realization of uncertainty.

F_{qk} = time of block q in workstation k $\forall q, k$

$x_{iqk} = \begin{cases} 1, & \text{operation } i \text{ is assigned to block } q \text{ in workstation } k \\ 0, & \text{otherwise} \end{cases} \quad \forall i, q, k$

Second-stage decision variables:

The second-stage decision variables depend on the realization of scenarios and represent the corresponding block times, workstation times, and cycle times.

CT_s = cycle time in scenario s

Constraints:

Assignment Constraints: (1) ensures that each operation is assigned to exactly one block and exactly one workstation.

$$\sum_{k=E_i}^{L_i} \sum_{q=1}^{m_0} x_{iqk} = 1 \quad \forall i \quad (1)$$

Workstation Opening Constraints: (2) ensures that a workstation is opened whenever at least one block is assigned to it.

$$\sum_{i=1}^n x_{iqk} \leq n_0 \quad \forall q, k = E_i, \dots, L_i \quad (2)$$

Precedence Constraints: (3) ensures that each operation can only be assigned only after all of its predecessor operations are completed in an earlier block.

$$x_{iqk} \leq \sum_{r=1}^{k-1} \sum_{h=1}^{m_0} x_{jhr} + \sum_{h=1}^{q-1} x_{jhk} \quad \forall i, q, \forall j \in P_i, k = \max(E_i; E_j), \dots, \min(L_i; L_j) \quad (3)$$

Not Same Block Constraints (4) ensures that operation pairs subject to block exclusion requirements are not assigned to the same block.

$$x_{jqk} \leq (1 - x_{iqk}) \quad \forall (i, j) \in EB, \forall q, k = \max(E_i; E_j), \dots, \min(L_i; L_j) \quad (4)$$

Not Same Workstation Constraints: (5) ensures that operation pairs subject to workstation exclusion requirements are not assigned to the same workstation.

$$\sum_{q=1}^{m_0} x_{jqk} \leq (1 - \sum_{q=1}^{m_0} x_{iqk}) \quad \forall (i, j) \in ES, k = \max(E_i; E_j), \dots, \min(L_i; L_j) \quad (5)$$

Same Workstation Constraints: (6) ensures that operation pairs subject to workstation inclusion requirements are assigned to the same workstation.

$$\sum_{k=E_j}^{L_j} \sum_{q=1}^{m_0} kx_{jqk} = \sum_{k=E_i}^{L_i} \sum_{q=1}^{m_0} kx_{iqk} \quad \forall (i, j) \in IS \quad (6)$$

Cycle Time Constraints: (7) and (8) ensure that block times are identified as the maximum operation time among the operations assigned to the block, workstation times are computed as the sum of the corresponding block times and the

maintenance time of the workstation, and the cycle time is equal to the maximum workstation time.

$$x_{iqk}t_i \leq F_{qk} \quad \forall i, q, \quad k = E_i, \dots, L_i \quad (7)$$

$$u_{ks}z_k + \sum_{q=1}^{m_0} F_{qk} \leq CT_s \quad \forall s, k \quad (8)$$

Sign Constraint: (9) represents the binary and non-negativity requirements.

$$x_{iqk} = \{0, 1\} \quad \forall i, q, \quad k = E_i, \dots, L_i \quad (9)$$

The objective function is to minimize expected cycle time and is expressed below.

$$\text{Min } \sum_{s=1}^S p_s CT_s \quad (10)$$

Note that the above SP model is a mixed integer linear program whose computational complexity increases with the number of operations, the number of workstations, the number of blocks per workstation, and the number of scenarios.

A numerical example:

We illustrate the problem with a sample instance taken from Belmokhtar et al. (2006) for machining line design of the manufacturing the mechanical desktop (depicted in the figure below).

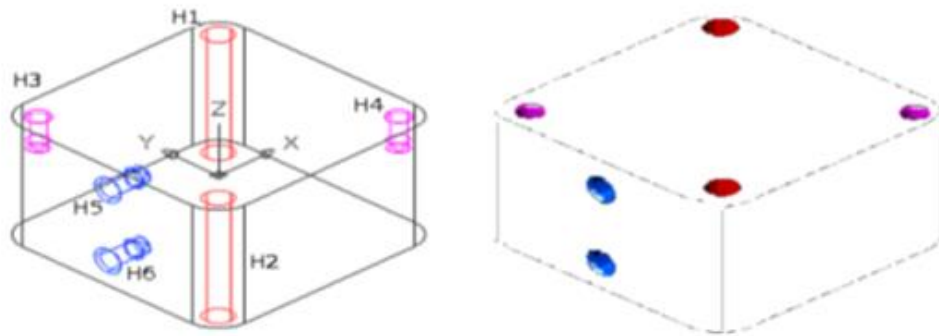


Figure 3: Body of a Mechanical Desktop (excerpted from Belmokhtar et al. 2006)

The instance represents a simplified real-life case in which the operations, precedence relations, and workstation exclusion and inclusion constraints are derived from real industrial settings.

They consider a Type-I TLBP with a predefined set of available blocks; therefore, they do not report the cycle time and individual operation times, which were only required for block formation. We generate the operation times data from $U[10, 20]$, following Guschinskaya and Dolgui (2009), who were inspired by real-life applications.

We report the real operations and their generated operation times in Table 6.

Table 6: Operations – descriptions and times

| Operation | Description | Time | Operation | Description | Time |
|-----------|----------------|------|-----------|----------------|------|
| 1 | Drill hole H1 | 16 | 8 | Ream hole H4 | 17 |
| 2 | Drill hole H2 | 15 | 9 | Drill hole H5 | 17 |
| 3 | Drill hole H3 | 14 | 10 | Countersink H5 | 16 |
| 4 | Countersink H3 | 19 | 11 | Tap a hole H5 | 20 |
| 5 | Ream hole H3 | 13 | 12 | Drill hole H6 | 10 |
| 6 | Drill hole H4 | 14 | 13 | Countersink H6 | 13 |
| 7 | Countersink H4 | 19 | 14 | Tap a hole H6 | 17 |

The set of precedence constraints is $\{(3; 4); (4; 5); (6; 7); (7; 8); (9; 10); (1; 11); (2; 11); (5; 11); (8; 11); (10; 11); (11; 13); (12; 13); (1; 14); (2; 14); (5; 14); (8; 14); (10; 14); (13; 14)\}$.

The set of workstation exclusion constraints is $\{(1; 5); (1; 8); (2; 5); (2; 8); (3; 5); (3; 8); (4; 5); (4; 8); (5; 10); (5; 12); (5; 13); (5; 6); (5; 7); (5; 9); (6; 8); (7; 8); (8; 10); (8; 12); (8; 13); (9; 11); (10; 11); (11; 12); (11; 13); (11; 14)\}$.

The set of workstation inclusion constraints is $\{(1; 2); (3; 6)\}$.

Belmokhtar et al. (2006) minimize total cost (with modified cost data), arriving at an optimal solution with 5 workstations ($m = 5$), maximum of 2 blocks per workstation ($m_o = 2$), and maximum of 4 operations per block ($n_o = 4$). These values are adopted as parameters in our model.

Table 7 presents the technological constraints, including operation times, immediate precedence relationships, and block and workstation exclusion constraints.

Table 7: Operation times and technological constraints

| Operation | Process Time | <i>EB</i> | <i>ES</i> | <i>IS</i> | <i>IP_i</i> |
|-----------|--------------|------------------------|------------------------|-----------|-----------------------|
| 1 | 16 | 5,8 | 5,8 | 2 | |
| 2 | 15 | 5,8 | 5,8 | 1 | |
| 3 | 14 | 5,8 | 5,8 | 6 | |
| 4 | 19 | 5,8 | 5,8 | | 3 |
| 5 | 13 | 1,2,3,4,6,7,9,10,12,13 | 1,2,3,4,6,7,9,10,12,13 | | 4 |
| 6 | 14 | 5,8 | 5,8 | 3 | |
| 7 | 19 | 5,8 | 5,8 | | 6 |
| 8 | 17 | 1,2,3,4,6,7,10,12,13 | 1,2,3,4,6,7,10,12,13 | | 7 |
| 9 | 17 | 5,11 | 5,11 | | |
| 10 | 16 | 5,8,11 | 5,8,11 | | 9 |
| 11 | 20 | 9,10,12,13,14 | 9,10,12,13,14 | | 1,2,5,10 |
| 12 | 10 | 5,8,11 | 5,8,11 | | |
| 13 | 13 | 5,8,11 | 5,8,11 | | 11,12 |
| 14 | 17 | 11 | 11 | | 1,2,5,8,10,13 |

The individual maintenance time of the workstations and the corresponding probabilities for each maintenance type are given in Table 8.

Table 8: Independent maintenance times and probabilities

| Type | Time of Maintenance | Probability (π_{kh}) |
|------|---------------------|----------------------------|
| 1 | 0 | 0.5 |
| 2 | 10 | 0.5 |

Table 9 enumerates the scenarios along with the corresponding maintenance times of the workstations and their associated probabilities.

Table 9: Scenarios and maintenance times with related probabilities

| Scenario <i>s</i> | Workstation / Maintenance Time | | | | | Probability <i>p_s</i> |
|----------------------|--------------------------------|-----------------------|-----------------------|-----------------------|-----------------------|-------------------------------------|
| | <i>u_{1s}</i> | <i>u_{2s}</i> | <i>u_{3s}</i> | <i>u_{4s}</i> | <i>u_{5s}</i> | |
| 1 | 0 | 0 | 0 | 0 | 0 | 0.03125 |
| 2 | 0 | 0 | 0 | 0 | 10 | 0.03125 |
| 3 | 0 | 0 | 0 | 10 | 0 | 0.03125 |
| 4 | 0 | 0 | 0 | 10 | 10 | 0.03125 |
| 5 | 0 | 0 | 10 | 0 | 0 | 0.03125 |
| 6 | 0 | 0 | 10 | 0 | 10 | 0.03125 |
| 7 | 0 | 0 | 10 | 10 | 0 | 0.03125 |
| 8 | 0 | 0 | 10 | 10 | 10 | 0.03125 |
| 9 | 0 | 10 | 0 | 0 | 0 | 0.03125 |

| Scenario S | Workstation / Maintenance Time | | | | | Probability P_s |
|-----------------|--------------------------------|----------|----------|----------|----------|----------------------|
| | u_{1s} | u_{2s} | u_{3s} | u_{4s} | u_{5s} | |
| 10 | 0 | 10 | 0 | 0 | 10 | 0.03125 |
| 11 | 0 | 10 | 0 | 10 | 0 | 0.03125 |
| 12 | 0 | 10 | 0 | 10 | 10 | 0.03125 |
| 13 | 0 | 10 | 10 | 0 | 0 | 0.03125 |
| 14 | 0 | 10 | 10 | 0 | 10 | 0.03125 |
| 15 | 0 | 10 | 10 | 10 | 0 | 0.03125 |
| 16 | 0 | 10 | 10 | 10 | 10 | 0.03125 |
| 17 | 10 | 0 | 0 | 0 | 0 | 0.03125 |
| 18 | 10 | 0 | 0 | 0 | 10 | 0.03125 |
| 19 | 10 | 0 | 0 | 10 | 0 | 0.03125 |
| 20 | 10 | 0 | 0 | 10 | 10 | 0.03125 |
| 21 | 10 | 0 | 10 | 0 | 0 | 0.03125 |
| 22 | 10 | 0 | 10 | 0 | 10 | 0.03125 |
| 23 | 10 | 0 | 10 | 10 | 0 | 0.03125 |
| 24 | 10 | 0 | 10 | 10 | 10 | 0.03125 |
| 25 | 10 | 10 | 0 | 0 | 0 | 0.03125 |
| 26 | 10 | 10 | 0 | 0 | 10 | 0.03125 |
| 27 | 10 | 10 | 0 | 10 | 0 | 0.03125 |
| 28 | 10 | 10 | 0 | 10 | 10 | 0.03125 |
| 29 | 10 | 10 | 10 | 0 | 0 | 0.03125 |
| 30 | 10 | 10 | 10 | 0 | 10 | 0.03125 |
| 31 | 10 | 10 | 10 | 10 | 0 | 0.03125 |
| 32 | 10 | 10 | 10 | 10 | 10 | 0.03125 |

Table 10 reports the ordered operation times, precedence relations, block and workstation exclusion constraints, and workstation inclusion constraints. All predecessor and all successor sets are constructed by considering the given precedence relations.

Table 10: Ordered operation times and technological constraints

| Operat ion | Process Time | EB | ES | IS | IP_i | P_i | S_i |
|---------------|-----------------|--------------------------|--------------------------|------|-------------------|-----------------------------------|------------------|
| 11 | 20 | 9,10,12,13,14 | 9,10,12,13,14 | | 1,2,5,10 | 1,2,3,4,5,6,7,8,9,10 | 13,14 |
| 7 | 19 | 5,8 | 5,8 | | 6 | 6 | 8,11,13, 14 |
| 4 | 19 | 5,8 | 5,8 | | 3 | 3 | 5,11,13, 14 |
| 14 | 17 | 11 | 11 | | 1,2,5,8,1 0,13 | 1,2,3,4,5,6,7,8,9,10, 11,12,13 | |
| 9 | 17 | 5,11 | 5,11 | | | | 10,11,13 ,14 |
| 8 | 17 | 1,2,3,4,6,7,10, 12,13 | 1,2,3,4,6,7,10, 12,13 | | 7 | 6,7 | 11,13,14 |
| 10 | 16 | 5,8,11 | 5,8,11 | | 9 | 9 | 11,13,14 |
| 1 | 16 | 5,8 | 5,8 | 2 | | | 11,13,14 |
| 2 | 15 | 5,8 | 5,8 | 1 | | | 11,13,14 |
| 6 | 14 | 5,8 | 5,8 | 3 | | | 7,8,11,1 3,14 |

| Operation | Process Time | EB | ES | IS | IP _i | P _i | S _i |
|-----------|--------------|------------------------|------------------------|----|-----------------|----------------------------|----------------|
| 3 | 14 | 5,8 | 5,8 | 6 | | | 4,5,11,13,14 |
| 13 | 13 | 5,8,11 | 5,8,11 | | 11,12 | 1,2,3,4,5,6,7,8,9,10,11,12 | 14 |
| 5 | 13 | 1,2,3,4,6,7,9,10,12,13 | 1,2,3,4,6,7,9,10,12,13 | | 4 | 3,4 | 11,13,14 |
| 12 | 10 | 5,8,11 | 5,8,11 | | | | 13,14 |

Table 11 presents the earliest and latest workstation values for each operation, which are obtained by using the complete predecessor and all successor sets.

Table 11: E_i and L_i values

| Operation | E _i | L _i |
|-----------|---|---|
| 1 | 1 | 5 |
| 2 | 1 | 5 |
| 3 | $\max\left\{\left\lceil\frac{\lceil\frac{ U_i \in IS_r P_i }{n_0} + 1}{m_0}\right\rceil, \left\lceil\frac{ U_i \in IS_r P_i UIS_r }{n_0 * m_0}\right\rceil\right\} =$ $\max\left\{\left\lceil\frac{0}{4} + 1\right\rceil, \left\lceil\frac{2}{2*4}\right\rceil\right\} = 1$ | $K - \max\left\{\left\lceil\frac{\lceil\frac{ U_i \in IS_r S_i }{n_0} + 1}{m_0}\right\rceil, \left\lceil\frac{ U_i \in IS_r S_i UIS_r }{n_0 * m_0}\right\rceil\right\} + 1 =$ $5 - \max\left\{\left\lceil\frac{5}{4} + 1\right\rceil, \left\lceil\frac{9}{2*4}\right\rceil\right\} + 1 = 4$ |
| 4 | $\left\lceil\frac{\lceil\frac{ P_i }{n_0} + 1}{m_0}\right\rceil = \left\lceil\frac{1}{4} + 1\right\rceil = 1$ | $K - \left\lceil\frac{\lceil\frac{ S_i }{n_0} + 1}{m_0}\right\rceil + 1 = 5 - \left\lceil\frac{4}{4} + 1\right\rceil + 1 = 5$ |
| 5 | 1 | 5 |
| 6 | 1 | 4 |
| 7 | 1 | 5 |
| 8 | 1 | 5 |
| 9 | 1 | 5 |
| 10 | 1 | 5 |
| 11 | 2 | 5 |
| 12 | 1 | 5 |
| 13 | 2 | 5 |
| 14 | 2 | 5 |

The SP model is solved, and the results are reported in Table 12.

Table 12: Assignments of operations and block times

| Workstation | 1 | 2 | 3 | 4 | 5 |
|-------------|----------|-----|--------|-----|----|
| Operations | 1,2,9,12 | 3,6 | 4,7,10 | 5,8 | 11 |
| Block Time | 17 | 14 | 19 | 17 | 20 |

The optimal objective function value is 38.25. This value is computed using Equation (10), with the detailed scenario-dependent cycle times and their corresponding probabilities reported in Table 13.

Table 13: Cycle time of scenarios with probabilities

| | | | | | | | | |
|--------------------------------------|---------|---------|---------|---------|---------|---------|---------|---------|
| Scenario, s | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Cycle Time, CT_s | 31 | 40 | 31 | 40 | 31 | 40 | 31 | 40 |
| Probability, p_s | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 |
| Scenario, s | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Cycle Time, CT_s | 31 | 40 | 31 | 40 | 31 | 40 | 31 | 40 |
| Probability, p_s | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 |
| Scenario, s | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| Cycle Time, CT_s | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 |
| Probability, p_s | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 |
| Scenario, s | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| Cycle Time, CT_s | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 |
| Probability, p_s | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 |

3.6 Constraint Programming Model

Constraint Programming (CP) is a modeling technique used to solve complex optimization problems by systematically searching for solutions that satisfy a set of constraints. It is particularly useful when dealing with problems that involve numerous variables and constraints, making Mixed Integer Linear Programming models inefficient. It allows the use of non-linear relations directly, eliminating the need for additional decision variables to linearize such expressions. Furthermore, CP formulations can naturally incorporate logical expressions, which enhance their modeling flexibility.

In this study, we construct a constraint programming (CP) model based on this scenario structure. The following parameters and variables are used to state the model.

Parameters:

n = number of operations

m = number of workstations

S = number of scenarios

n_o = maximum number of operations per block

m_o = maximum number of blocks per workstation

t_i = processing (operation) time of operation $i, i = 1, \dots, n$

P_i = set of all predecessors of operation $i, i = 1, \dots, n$

ES = set of operation pairs that cannot be assigned to the same workstation

EB = set of operation pairs that cannot be assigned to the same block

IS = set of operation pairs that should be assigned to the same workstation

E_i = the earliest workstation that operation i can be assigned, $i = 1, \dots, n$

L_i = the latest workstation that operation i can be assigned, $i = 1, \dots, n$

u_{ks} = maintenance time of workstation k in scenario $s, s = 1, \dots, S, k = 1, \dots, m$

p_s = realization probability of scenario $s, s = 1, \dots, S$

First-stage decision variables:

F_{qk} = time of block q in workstation k $\forall q, k$

$x_i = k$, if operation i is assigned to workstation k $\forall i, k$

$y_i = q$, if operation i is assigned to block q $\forall i, q$

Second-stage decision variables:

CT_s = cycle time in scenario s

Constraints:

Operation Limitation Constraints: (11) ensures that the number of operations assigned to a block does not exceed the maximum allowable number of operations per block.

$$\sum_{i|L_i \geq k \geq E_i} (x_i = k \ \& \ y_i = q) \leq n_o \quad \forall q, k \quad (11)$$

Precedence Constraints: (12) ensures that each operation can only be assigned after all of its predecessor operations are completed in an earlier block.

$$x_j \leq x_i \quad \forall i, j \in P_i \quad (12)$$

$$y_j < y_i \text{ where } x_i = x_j \quad \forall i, j \in P_i$$

Not Same Block Constraints (13) ensures that operation pairs subject to block exclusion requirements are not assigned to the same block.

$$y_j \neq y_i \text{ where } x_i = x_j \quad \forall (i, j) \in EB \quad (13)$$

Not Same Workstation Constraints: (14) ensures that operation pairs subject to workstation exclusion requirements are not assigned to the same workstation.

$$x_i \neq x_j \quad \forall (i, j) \in ES \quad (14)$$

Same Workstation Constraints: (15) ensures that operation pairs subject to workstation inclusion requirements are assigned to the same workstation.

$$x_i = x_j \quad \forall (i, j) \in IS \quad (15)$$

Workstation Limitation Constraints: (16) and (17) ensure that each operation is assigned to a feasible workstation within its allowable range

$$x_i \geq E_i \quad \forall i \quad (16)$$

$$x_i \leq L_i \quad \forall i \quad (17)$$

Cycle Time Constraints: (18) and (19) ensure that block times are defined as the maximum processing time among the operations assigned to the block, workstation times are computed as the sum of the corresponding block times and the maintenance time of the workstation, and the cycle time is equal to the maximum workstation time.

$$(x_i = k \ \& \ y_i = q) t_i \leq F_{qk} \quad \forall q, k, \ \forall i | E_i \leq k \leq L_i \quad (18)$$

$$u_{ks}(x_i = k) + \sum_{q=1}^{m_0} F_{qk} \leq CT_s \quad \forall s, k, \ \forall i | E_i \leq k \leq L_i \quad (19)$$

Sign Constraints: (20) and (21) specify the binary and non-negativity requirements of the decision variables.

$$m \geq x_i \geq 1 \quad \forall i \quad (20)$$

$$m_o \geq y_i \geq 1 \quad \forall k \quad (21)$$

The objective function is to minimize expected cycle time and is expressed below.

$$\text{Min } \sum_{s=1}^S p_s CT_s \quad (22)$$

3.7 Expected Value Problem and the Value of the Stochastic Solution

The Expected Value (EV) problem is constructed by replacing the random parameters with their expected values, thereby yielding the deterministic counterpart of the stochastic program. In this formulation, the random parameters are substituted by their expected values, resulting in a single scenario problem, which is referred to as the expected value scenario.

The EV model is significantly simpler, since the number of scenarios is reduced to one, and this simplification directly affects the computational complexity of the problem.

We formulate the EV model by replacing Constraint (8), i.e.,

$$u_{ks} + \sum_q F_{qk} \leq CT_s \quad \forall s, k$$

with

$$\bar{u}_k + \sum_q F_{qk} \leq CT \quad \forall k \quad (23)$$

$$\text{where } \bar{u}_k = \sum_{h=1}^3 \pi_{kh} \cdot u_{kh}$$

The objective function then becomes

$$\text{Min } CT \quad (24)$$

The EV model is formulated as follows

$$\text{Min } CT$$

subject to constraints (1) – (7), (9) and (23).

The objective function expressed in (24) represents the cycle time of the EV scenario. The expected cycle time of the EV solution, denoted by EEV , is obtained by fixing the first-stage decision variable (the value of x_{iqk}), to their optimal EV values and then computing the optimal second-stage decisions using Constraints (26) – (27), without explicitly solving the full linear program.

$$\text{Min } z = \sum_s p_s CT_s \quad (25)$$

subject to

$$x^*_{iqk} t_i \leq \sum_q F_{qk} \quad \forall i, q, k \quad (26)$$

$$u_{ks} + \sum_q F_{qk} \leq CT_s \quad \forall s, k \quad (27)$$

Note that the remaining constraints are implicitly satisfied by the EV model, since the first-stage decisions are fixed to the values obtained from the EV solution.

Let Z^*_{EEV} denote the optimal objective function value of the above model, and let Z^*_{SP} denote the optimal objective function value of the original stochastic program.

The numerical example (cont.)

The calculated average maintenance times are presented in Table 14.

Table 14: Average maintenance times

| Workstation | Case | Maintenance Time | Probability | Average Maintenance Time |
|-------------|------|------------------|-------------|--------------------------|
| 1 | 1 | 0 | 0.5 | 5 |
| 1 | 2 | 10 | 0.5 | |
| 2 | 1 | 0 | 0.5 | 5 |
| 2 | 2 | 10 | 0.5 | |
| 3 | 1 | 0 | 0.5 | 5 |
| 3 | 2 | 10 | 0.5 | |
| 4 | 1 | 0 | 0.5 | 5 |
| 4 | 2 | 10 | 0.5 | |
| 5 | 1 | 0 | 0.5 | 5 |
| 5 | 2 | 10 | 0.5 | |

The EV model is solved, and the results are reported in Table 15.

Table 15: Assignments of operations and block times

| Workstation | 1 | | 2 | 3 | | 4 | | 5 | |
|-------------|-----|-----|-----|------|------|----|----|----|----|
| Operations | 3,6 | 4,7 | 8,9 | 2,10 | 1,12 | 5 | 11 | 13 | 14 |
| Block Time | 14 | 19 | 17 | 16 | 16 | 13 | 20 | 13 | 17 |

Using the assignments reported in Table 15, the EEV model is solved. The following constraints are added to the SP model.

$$\begin{aligned}
 x_{123} = 1 & & x_{421} = 1 & & x_{721} = 1 & & x_{1013} = 1 & & x_{1315} = 1 \\
 x_{213} = 1 & & x_{514} = 1 & & x_{822} = 1 & & x_{1124} = 1 & & x_{1425} = 1 \\
 x_{311} = 1 & & x_{611} = 1 & & x_{922} = 1 & & x_{1223} = 1 & &
 \end{aligned}$$

The corresponding objective function value is 42.0625. This value is computed using Equation (25) with the detailed scenario-dependent cycle times and their associated probabilities reported in Table 16.

Table 16: Cycle time for each scenario with probabilities

| | | | | | | | | |
|--------------------|---------|---------|---------|---------|---------|---------|---------|---------|
| Scenario, s | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Cycle Time, CT_s | 33 | 43 | 33 | 43 | 42 | 43 | 42 | 43 |
| Probability, p_s | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 |
| Scenario, s | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Cycle Time, CT_s | 43 | 43 | 43 | 43 | 43 | 43 | 43 | 43 |
| Probability, p_s | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 |
| Scenario, s | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| Cycle Time, CT_s | 40 | 43 | 40 | 43 | 42 | 43 | 42 | 43 |
| Probability, p_s | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 |
| Scenario, s | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| Cycle Time, CT_s | 43 | 43 | 43 | 43 | 43 | 43 | 43 | 43 |
| Probability, p_s | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 | 0.03125 |

Thus, $z^*_{EEV} = \sum_s p_s CT_s = 42.0625$ and $z^*_{SP} = \sum_{s=1}^S p_s CT_s = 41.8125$.

If z^*_{SP} is close to z^*_{EEV} , then the EV solution leads to a near optimal result. In such cases, the EV solution may be preferred, as it can be obtained much more easily by solving a single-scenario problem instead of a multi-scenario stochastic program, without sacrificing solution quality. On the other hand, if z^*_{EEV} is significantly larger than z^*_{SP} , then the stochastic program becomes more valuable, and it is worthwhile to seek its optimal solution.

The Value of Stochastic Solution (VSS) formalizes this concept by quantifying the benefit of explicitly considering uncertainty. It represents the improvement in expected cycle time achieved by solving the stochastic program rather than its deterministic expected value counterpart.

VSS is computed as follows:

$$VSS = z^*_{EEV} - z^*_{SP}$$

VSS can also be computed as a relative measure as:

$$VSS = z^*_{SP} / z^*_{EEV}$$

$1 - \frac{z^*_{SP}}{z^*_{EEV}}$ can be interpreted as the percentage improvement obtained by using the stochastic model instead of the EV model.

The numerical example (cont.)

Value of the stochastic solution for the illustrative example can be measured, both absolute and relative, as stated below:

$$VSS = z^*_{EEV} - z^*_{SP} = 42.0625 - 41.8125 = 0,25$$

$$VSS = \frac{z^*_{SP}}{z^*_{EEV}} = \frac{41.8125}{42.0625} = 0.994 = 99\%$$

These results indicate that the stochastic solution provides 1% improvement over the expected value solution.

CHAPTER 4

BRANCH AND BOUND ALGORITHM

Mathematical models can efficiently solve small and medium-sized problem instances to optimality. However, as the problem size increases, the resulting formulations become computationally challenging due to the rapid growth in the number of variables and constraints. To address larger instances, we therefore propose Branch and Bound (B&B) algorithm as an alternative exact solution approach. The efficiency of the proposed algorithm is enhanced through effective reduction and bounding mechanisms.

The B&B algorithm assigns operations to blocks and blocks to workstations sequentially, starting from the first block of the first workstation.

4.1 Branching Scheme

We first index the operations in non-increasing order of their processing times, i.e., $t_1 \geq t_2 \geq \dots \geq t_n$.

Partial solutions of the branch and bound tree are generated starting from the first operation of the first block of the first workstation. For a given partial solution corresponding to the t^{th} block of k^{th} workstation, we either assign an operation to the current block or close the current block and open a new block either in the same workstation or in the next (k^{th} or $(k + 1)^{st}$) workstation.

While adding an operation to the current block, we only consider operations with higher indices than the last assigned operation. This strategy prevents duplication of partial solutions. Since branching always proceeds toward higher indices, the

first assigned operation, i.e., the longest operation according to the indexing rule, determines the cycle time of the block.

According to this branching scheme, a block is never closed if there exists an unassigned eligible operation with a higher index. An operation i is said to be eligible for assignment to the current block if its assignment does not violate any precedence relations or operational limitations, such that:

- i. all predecessors of operation i are already assigned to earlier blocks,
- ii. no operation j , such that $(i, j) \in EB$ is assigned to the current block,
- iii. no operation j , such that $(i, j) \in ES$ is assigned to the current workstation,
- iv. the current workstation index lies between the earliest and latest feasible workstation indices of operation, i.e., E_i and L_i .

If no eligible operation exists, or if the maximum number of operations per block (n_o) is reached, the current block is closed. Once a block is closed, either a new block is opened in the same workstation or a new block is opened in the next workstation, which implies closing the current workstation.

An operation i is eligible for assignment to block $(t + 1)^{st}$ of the current workstation if:

- i. all predecessors of i are already assigned
- ii. no operation j such that $(i, j) \in ES$ is assigned to the current workstation

The current workstation is closed if

$$\sum_s p_s (W_k + \min_{i \in EL_k} \{t_i\} + u_{ks}) \geq UB$$

where

W_k = current workload of workstation k

EL_k = set of eligible operations for workstation k

UB = best known objective function value

The current workstation is not closed if at least one of the following conditions holds:

- i. there exists an unassigned eligible operation i such that
$$W_k + t_i + u_{ks} \leq \max\left\{\max_{v=1,\dots,k-1} W_v + u_{vs}, LB_s\right\} \quad \forall s$$

$$LB_s = \text{a valid lower bound for } CT \text{ of scenario } s$$
- ii. there exists an unassigned eligible operation i such that $L_i = k$ where k is the current workstation

The current workstation is closed if one of the following conditions holds:

- i. there exist no feasible operations
- ii. the maximum numbers of operations per block, n_o , and blocks per workstation, m_o , are reached
- iii. assigning an eligible operation increases the cycle time beyond UB

The following additional fathoming rules are applied based on remaining capacity and operational limits:

- For each node, we check whether the remaining capacity of the workstation is sufficient for the remaining unassigned eligible operations. If not, the node is fathomed.
- Moreover, for each workstation, IS_r sets are examined. If the cardinality of such a set is smaller than the remaining capacity, the node is fathomed; if it is equal, the operations are assigned simultaneously without further branching.

When closing the last workstation, the node is fathomed if there exist two operations i and j such that $(i, j) \in ES$ and neither is assigned.

The values of E_i and L_i are updated when a workstation is closed, taking into account the remaining operations and their relations.

An initial upper bound is provided fast and it is updated whenever a complete solution with a better objective function value is found in our B&B search.

Nodes are selected according to their depth in the branch and bound tree and their lower bound values. Nodes at deeper levels with smaller lower bounds are explored first. A node is fathomed whenever its lower bound exceeds the current upper bound or when insufficient capacity remains for unassigned operations.

4.2 Lower Bounds

Lower bounds are used when closing workstations. Once a workstation is closed, the problem is reduced to a smaller set of operations and fewer workstations. Node selection is based on lower bound values, prioritizing nodes with smaller bounds in order to reach high-quality feasible solutions earlier.

Telemeci and Azizoglu (2024) propose a group-forming approach for the deterministic TLBP. In their method, operations are grouped by ignoring precedence relations and then treated as aggregated operations assigned to workstations, without considering the number of blocks per workstation constraint.

In this study, we enhance their group-forming procedure and compute a lower bound for each scenario s , LB_s . The overall lower bound is defined as $\sum_s p_s LB_s$.

A group is defined as a set of operations that can be processed within the same block. Two operations can belong to the same group if there are no precedence relations, block exclusion constraints, or workstation exclusion constraints between them. Such operations are referred to as linked operations.

We use the following procedure to form groups.

Group Forming Procedure

Step 1. Order the operations in non-increasing order of processing times,

$$\text{i.e., } t_1 \geq t_2 \geq \dots \geq t_n$$

Step 2. Starting from the first operation form groups as follows:

- Assign the first unassigned operation to a group that already contains at least one linked operation.
- If no such operation exists, open a new group

Step 3. Let r_g denote the number of operations in group g

For group g , form $\left\lceil \frac{r_g}{n_0} \right\rceil$ blocks by assigning the first n_0 operations to the first block, the next n_0 operations to the second block and so on.

The total number of blocks is denoted by $B = \sum_g \left\lceil \frac{r_g}{n_0} \right\rceil$. The processing time of each block is determined by the first, hence longest, operation assigned to that block.

Let tt_b be the processing time of block b where $b = 1, 2, \dots, \sum_g \left\lceil \frac{r_g}{n_0} \right\rceil$

The numerical example (cont.)

Table 17 presents the ordered operation processing times together with precedence relations, block/workstation exclusion constraints, and workstation inclusion constraints. For each operation, predecessor and successor sets are constructed based on the given precedence relations.

Table 17: Ordered operation times and technological constraints

| Operation | Process Time | EB | ES | IS | P_i |
|-----------|--------------|------------------------|------------------------|------|-------------------------------|
| 11 | 20 | 9,10,12,13,14 | 9,10,12,13,14 | | 1,2,3,4,5,6,7,8,9,10 |
| 7 | 19 | 5,8 | 5,8 | | 6 |
| 4 | 19 | 5,8 | 5,8 | | 3 |
| 14 | 17 | 11 | 11 | | 1,2,3,4,5,6,7,8,9,10,11,12,13 |
| 9 | 17 | 5,11 | 5,11 | | |
| 8 | 17 | 1,2,3,4,6,7,10,12,13 | 1,2,3,4,6,7,10,12,13 | | 6,7 |
| 10 | 16 | 5,8,11 | 5,8,11 | | 9 |
| 1 | 16 | 5,8 | 5,8 | 2 | |
| 2 | 15 | 5,8 | 5,8 | 1 | |
| 6 | 14 | 5,8 | 5,8 | 3 | |
| 3 | 14 | 5,8 | 5,8 | 6 | |
| 13 | 13 | 5,8,11 | 5,8,11 | | 1,2,3,4,5,6,7,8,9,10,11,12 |
| 5 | 13 | 1,2,3,4,6,7,9,10,12,13 | 1,2,3,4,6,7,9,10,12,13 | | 3,4 |
| 12 | 10 | 5,8,11 | 5,8,11 | | |

Based on the relationships outlined in Table 17, the operations are partitioned into groups using the proposed group forming procedure. The process starts with the operation having the largest processing time, which is Operation 11, and this operation is assigned to Group 1. The next operation in descending order, Operation 7, cannot be assigned to Group 1 because it precedes Operation 11; therefore, it is assigned to Group 2. Subsequently, Operation 4 is also assigned to Group 2. Operation 14 cannot be assigned to either Group 1 or Group 2: it cannot be grouped with Operation 11 due to block exclusion constraints, nor with Operation 7 due to precedence relations. This process continues in this manner until all operations are assigned to exactly one group.

Figure 4 illustrates the resulting network representation, where nodes correspond to operations and connected components represent groups formed by linked operations.

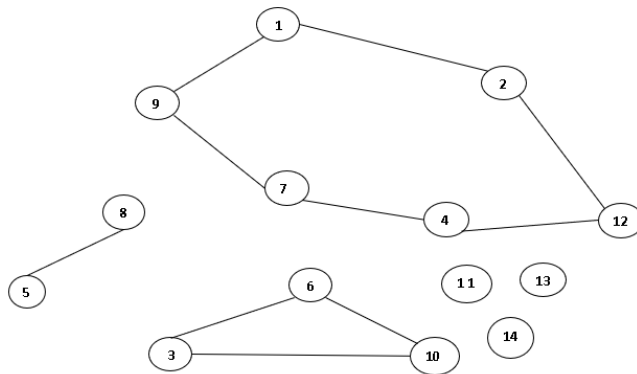


Figure 4: Connected subgraph of operations

The resulting six groups and corresponding operations are presented in Table 18.

Table 18: Operation assignments to groups

| Group1 | Group2 | Group3 | Group4 | Group5 | Group6 |
|--------|--------|--------|--------|--------|--------|
| 11 | 7 | 14 | 8 | 10 | 13 |
| | 4 | | 5 | 6 | |
| | 9 | | | 3 | |
| | 1 | | | | |
| | 2 | | | | |
| | 12 | | | | |

Note that all groups, except Group 2 contain no more than n_0 operations and therefore form a single block. For Group 2, $\left\lceil \frac{r_g}{n_0} \right\rceil = \left\lceil \frac{6}{4} \right\rceil = 2$, hence it produces two blocks. As a result, a total of seven blocks is obtained from the six groups.

The operations assigned to each block and their corresponding processing times are shown in Table 19.

Table 19: Assignments of operations and block times

| Block | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------------|----|------------|-------|----|------|----------|----|
| Operations | 11 | 7, 4, 9, 1 | 2, 12 | 14 | 8, 5 | 10, 6, 3 | 13 |
| Block Time | 20 | 19 | 15 | 17 | 17 | 16 | 13 |

Lower Bound Computations:

The first lower bound is obtained by treating each block as an aggregated operation and using its block processing time as the corresponding operation time. Accordingly, the resulting LB , for scenario s denoted by LB_{1s} is defined as:

$$LB_{1s} = \left\lceil \frac{\sum_b tt_b + \sum_k u_{ks}}{m} \right\rceil$$

As in Telemeci and Azizoglu (2024), this bound adapts the lower bound developed for the single Type-II Assembly Line Balancing Problem. In this study, the bound is further strengthened by allowing operation splitting while preserving maintenance times. The resulting modified bound $L\hat{B}_{1s}$ is expressed as:

$$L\hat{B}_{1s} = \left\lceil \max_k \{u_{ks}\} + \max \left\{ 0, \frac{\sum_b tt_b - [\max_k \{u_{ks}\} m - \sum_k u_{ks}]}{m} \right\} \right\rceil$$

The second lower bound is extended directly from Klein and Scholl (1996). Maintenance times, as well as block and workstation-related information, are incorporated into the formulation to obtain the second lower bound, denoted by LB_{2s} .

$$LB_{2s} = \max_i \left\{ t_i + \min_{E_i \leq k \leq L_i} \{u_{ks}\} \right\}$$

The lower bound for each scenario s , LB_s is computed as:

$$\max \{L\hat{B}_{1s}, LB_{2s}\}$$

Finally, the global lower bound over all scenarios is defined as:

$$\sum_S p_S LB_S.$$

Table 20 reports the individual lower bounds for each scenario as well as the resulting global lower bound, which is calculated as 29 for the illustrative example.

Table 20: Lower Bounds

| Scenario | $LB_{s_i} = \lfloor \frac{117r_i + 24r_{i+1}}{5} \rfloor$ | $LB_{s_i} = \lfloor \frac{117r_i + 24r_{i+1}}{5} \rfloor$ | $LB_{s_i} = \lfloor \frac{117r_i + 24r_{i+1}}{5} \rfloor$ | $LB_{s_i} = \max(r_i + \frac{\min(r_{i+1}, r_{i+2})}{5})$ | $\max(LB_{s_i}, LB_{s_j})$ | |
|----------|---|--|---|---|----------------------------|----|
| 1 | $\lfloor \frac{117+0}{5} \rfloor = 24$ | $LB_{s_1} = 0 + \max(0, \frac{117-(0+5-0)}{5}) = 24$ | | | $LB_{s_{11}} = 20$ | 24 |
| 2 | $\lfloor \frac{117+20}{5} \rfloor = 26$ | $LB_{s_2} = 10 + \max(0, \frac{117-(10+5-10)}{5}) = 26$ | | | $LB_{s_{22}} = 20$ | 26 |
| 3 | $\lfloor \frac{117+40}{5} \rfloor = 28$ | $LB_{s_3} = 10 + \max(0, \frac{117-(10+5-10)}{5}) = 26$ | | | $LB_{s_{33}} = 20$ | 26 |
| 4 | $\lfloor \frac{117+60}{5} \rfloor = 28$ | $LB_{s_4} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{44}} = 20$ | 28 |
| 5 | $\lfloor \frac{117+80}{5} \rfloor = 28$ | $LB_{s_5} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{55}} = 20$ | 28 |
| 6 | $\lfloor \frac{117+100}{5} \rfloor = 28$ | $LB_{s_6} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{66}} = 20$ | 28 |
| 7 | $\lfloor \frac{117+120}{5} \rfloor = 30$ | $LB_{s_7} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{77}} = 20$ | 28 |
| 8 | $\lfloor \frac{117+140}{5} \rfloor = 30$ | $LB_{s_8} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{88}} = 20$ | 30 |
| 9 | $\lfloor \frac{117+160}{5} \rfloor = 26$ | $LB_{s_9} = 10 + \max(0, \frac{117-(10+5-10)}{5}) = 26$ | | | $LB_{s_{99}} = 20$ | 26 |
| 10 | $\lfloor \frac{117+180}{5} \rfloor = 28$ | $LB_{s_{10}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{100}} = 20$ | 28 |
| 11 | $\lfloor \frac{117+200}{5} \rfloor = 28$ | $LB_{s_{11}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{111}} = 20$ | 28 |
| 12 | $\lfloor \frac{117+220}{5} \rfloor = 30$ | $LB_{s_{12}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{112}} = 20$ | 30 |
| 13 | $\lfloor \frac{117+240}{5} \rfloor = 28$ | $LB_{s_{13}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{113}} = 20$ | 28 |
| 14 | $\lfloor \frac{117+260}{5} \rfloor = 30$ | $LB_{s_{14}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{114}} = 20$ | 30 |
| 15 | $\lfloor \frac{117+280}{5} \rfloor = 30$ | $LB_{s_{15}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{115}} = 20$ | 30 |
| 16 | $\lfloor \frac{117+300}{5} \rfloor = 32$ | $LB_{s_{16}} = 10 + \max(0, \frac{117-(10+5-40)}{5}) = 32$ | | | $LB_{s_{116}} = 20$ | 32 |
| 17 | $\lfloor \frac{117+320}{5} \rfloor = 28$ | $LB_{s_{17}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{117}} = 20$ | 26 |
| 18 | $\lfloor \frac{117+340}{5} \rfloor = 28$ | $LB_{s_{18}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{118}} = 20$ | 28 |
| 19 | $\lfloor \frac{117+360}{5} \rfloor = 28$ | $LB_{s_{19}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{119}} = 20$ | 28 |
| 20 | $\lfloor \frac{117+380}{5} \rfloor = 30$ | $LB_{s_{20}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{200}} = 20$ | 30 |
| 21 | $\lfloor \frac{117+400}{5} \rfloor = 28$ | $LB_{s_{21}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{211}} = 20$ | 28 |
| 22 | $\lfloor \frac{117+420}{5} \rfloor = 30$ | $LB_{s_{22}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{222}} = 20$ | 30 |
| 23 | $\lfloor \frac{117+440}{5} \rfloor = 30$ | $LB_{s_{23}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{223}} = 20$ | 30 |
| 24 | $\lfloor \frac{117+460}{5} \rfloor = 32$ | $LB_{s_{24}} = 10 + \max(0, \frac{117-(10+5-40)}{5}) = 32$ | | | $LB_{s_{224}} = 20$ | 32 |
| 25 | $\lfloor \frac{117+480}{5} \rfloor = 28$ | $LB_{s_{25}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{225}} = 20$ | 28 |
| 26 | $\lfloor \frac{117+500}{5} \rfloor = 30$ | $LB_{s_{26}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{226}} = 20$ | 30 |
| 27 | $\lfloor \frac{117+520}{5} \rfloor = 30$ | $LB_{s_{27}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{227}} = 20$ | 30 |
| 28 | $\lfloor \frac{117+540}{5} \rfloor = 32$ | $LB_{s_{28}} = 10 + \max(0, \frac{117-(10+5-40)}{5}) = 32$ | | | $LB_{s_{228}} = 20$ | 32 |
| 29 | $\lfloor \frac{117+560}{5} \rfloor = 30$ | $LB_{s_{29}} = 10 + \max(0, \frac{117-(10+5-20)}{5}) = 26$ | | | $LB_{s_{229}} = 20$ | 30 |
| 30 | $\lfloor \frac{117+580}{5} \rfloor = 32$ | $LB_{s_{30}} = 10 + \max(0, \frac{117-(10+5-40)}{5}) = 32$ | | | $LB_{s_{230}} = 20$ | 32 |
| 31 | $\lfloor \frac{117+600}{5} \rfloor = 32$ | $LB_{s_{31}} = 10 + \max(0, \frac{117-(10+5-40)}{5}) = 32$ | | | $LB_{s_{231}} = 20$ | 32 |
| 32 | $\lfloor \frac{117+620}{5} \rfloor = 34$ | $LB_{s_{32}} = 10 + \max(0, \frac{117-(10+5-60)}{5}) = 34$ | | | $LB_{s_{232}} = 30$ | 34 |
| | | | | | $\sum p_i \cdot LB_i$ | 29 |

As stated previously, a node is fathomed whenever its lower bound exceeds the best-known objective function value, i.e., when $LB \geq UB$ where UB is the best-known objective function value. The proposed B&B algorithm employs a depth-first search strategy guided by lower bound values, where nodes at deeper levels of the tree are explored first. Once all nodes at a given branch are fathomed, the algorithm backtracks.

This strategy is preferred due to its relatively low memory requirements and its effectiveness in identifying high quality feasible solutions early in the search process.

CHAPTER 5

HEURISTIC APPROACHES

Although the mathematical models and the Branch and Bound (B&B) algorithm guarantee optimality through an exhaustive exploration of the solution space, their computational burden becomes prohibitive as the problem size increases. In particular, large-sized instances cannot be solved efficiently within reasonable time limits using exact approaches. Motivated by this limitation, we develop several heuristic approaches that aim to balance computational efficiency with acceptable solution quality.

Specifically, three heuristic methods are proposed:

- B&B-based Heuristic
- Model-based Heuristics
- Rule-based Heuristic

These approaches can be employed either independently or in combination, providing a flexible and practical tool for solving large scale optimization problems. Overall, they offer a promising alternative to exact methods by delivering high quality solutions within acceptable computational times.

5.1 Alpha Percent Heuristic – Branch and Bound-based

The Alpha Percent Heuristic follows the fundamental principles of Branch and Bound (B&B) algorithm, while integrating heuristic strategies aimed at balancing computational efficiency with solution quality. The primary objective is to effectively manage the trade-off between optimality and runtime, by selectively exploring the most promising regions of the solution space.

The Alpha Percent Heuristic introduces an innovative relaxation mechanism by modifying the bounding rules within the standard B&B algorithm framework. By permitting a controlled level of deviation from the strict bounding conditions, this method enables the exploration of a broader set of near optimal solutions in a fraction of the time required for exact optimization. The "alpha" parameter acts as a tunable control that adjusts the balance between exploration and precision, making the heuristic adaptable to a wide range of problem complexities and user defined solution thresholds.

As mentioned earlier, in the standard B&B algorithm, a node is fathomed when the lower bound LB exceeds or equals the upper bound UB , i. e., $LB \geq UB$. In the Alpha Percent Heuristic, however, the node is fathomed when the condition $(1 + \alpha)LB \geq UB$.

Note that, the parameter α controls the level of deviation. When α is small, the CPU times are significantly higher, whereas the larger values of α reduce the CPU times, though at the cost of increased deviation. In our experiments, we tested different values of α and found that $\alpha = 0.25$ offers a satisfactory trade-off between CPU time and deviation.

5.2 Matheuristics – Mathematical Model-based Heuristics

In addition to the B&B-based approach, we propose matheuristics that combines mathematical programming with heuristic improvement mechanisms. This approach consists of two sequential phases:

Phase 1: Constructing a feasible solution

Phase 2: Improving the solution returned by Phase 1

In the construction phase, a feasible solution is generated by solving simplified mathematical models. Subsequently, in the improvement phase, the quality of the

obtained solution is enhanced using two complementary strategies: simple pairwise interchange moves and optimization through a mathematical model.

Phase 1: Constructing a feasible solution

Model SP simultaneously forms blocks and assigns them to workstations, which significantly increases the difficulty of finding optimal solutions, especially under multiple scenarios. To address these challenges, the proposed heuristic decomposes the decision process into a sequence of more tractable stages.

In the first stage, operations are assigned to blocks while considering precedence relations, block exclusion constraints, workstation exclusion constraints, and limits on the number of operations per block. This assignment is performed using a single-scenario mathematical model with a simplified objective function.

In the second stage, the blocks obtained in the first stage are treated as aggregated entities (operations) and assigned to workstations using another mathematical model. During this assignment, updated precedence relations as well as workstation inclusion and exclusion constraints are taken into account. Consequently, the resulting model can be interpreted as a multi scenario assembly line balancing problem with additional side constraints. The workstation inclusion and exclusion constraints are analogous to the well-known zoning marriage and zoning divorce constraints, respectively.

5.2.1 Stage 1 – Block Formation Models

To generate alternative block structures, we employ four different mathematical models based on a single-scenario version of the original SP model. These models differ only in their objective functions, while sharing the same set of parameters and constraints. For completeness, the parameters, decision variables and constraints are restated below.

Parameters:

n = number of operations

m = number of workstations

n_o = max number of operations per block

m_o = max number of blocks per workstation

t_i = processing (operation) time of operation i , $i = 1, \dots, n$

P_i = set of all predecessors of operation i , $i = 1, \dots, n$

ES = set of operation pairs that cannot be assigned to the same workstation

EB = set of operation pairs that cannot be assigned to the same block

IS = set of operation pairs that should be assigned to the same workstation

E_i = the earliest workstation that operation i can be assigned, $i = 1, \dots, n$

L_i = the latest workstation that operation i can be assigned, $i = 1, \dots, n$

Decision variables:

F_{qk} = time of block q in workstation $k \quad \forall q, k$

$x_{iqk} = \begin{cases} 1, & \text{operation } i \text{ is assigned to block } q \text{ in workstation } k \\ 0, & \text{otherwise} \end{cases} \quad \forall i, q, k$

$y_{qk} = \begin{cases} 1, & \text{block } q \text{ in workstation } k \text{ is opened} \\ 0, & \text{otherwise} \end{cases} \quad \forall q, k$

Constraints:

Assignment Constraints: (28) ensures that each operation is assigned to exactly one block and exactly one workstation.

$$\sum_{k=E_i}^{L_i} \sum_{q=1}^{m_o} x_{iqk} = 1 \quad \forall i \quad (28)$$

Workstation Opening Constraints: (29a) and (29b) ensure that a workstation is opened whenever at least one block is assigned to it. Constraint (29a) enforces an upper bound on the number of operations that can be assigned to a block workstation pair. When the binary variables y_{qk} are explicitly included in the

objective function, constraint (29b) replaces (29a) to link block assignment decisions directly to workstation opening decisions.

$$\sum_{i=1}^n x_{iqk} \leq n_0 \quad \forall q, \quad k = E_i, \dots, L_i \quad (29a)$$

$$\sum_{i=1}^n x_{iqk} \leq n_0 * y_{qk} \quad \forall q, \quad k = E_i, \dots, L_i \quad (29b)$$

Precedence Constraints: (30) ensures that each operation can only be assigned only after all of its predecessor operations are completed in an earlier block.

$$x_{iqk} \leq \sum_{r=1}^{k-1} \sum_{h=1}^{m_0} x_{jhr} + \sum_{h=1}^{q-1} x_{jhk} \quad \forall i, q, \quad \forall j \in P_i, \quad k = \max(E_i; E_j), \dots, \min(L_i; L_j) \quad (30)$$

Not Same Block Constraints (31) ensures that operation pairs subject to block exclusion requirements are not assigned to the same block.

$$x_{jqk} \leq (1 - x_{iqk}) \quad \forall (i, j) \in EB, \forall q, \quad k = \max(E_i; E_j), \dots, \min(L_i; L_j) \quad (31)$$

Not Same Workstation Constraints: (32) ensures that operation pairs subject to workstation exclusion requirements are not assigned to the same workstation.

$$\sum_{q=1}^{m_0} x_{jqk} \leq (1 - \sum_{q=1}^{m_0} x_{iqk}) \quad \forall (i, j) \in ES, k = \max(E_i; E_j), \dots, \min(L_i; L_j) \quad (32)$$

Same Workstation Constraints: (33) ensures that operation pairs that operation pairs subject to workstation inclusion requirements are assigned to the same workstation.

$$\sum_{k=E_j}^{L_j} \sum_{q=1}^{m_0} k x_{jqk} = \sum_{k=E_i}^{L_i} \sum_{q=1}^{m_0} k x_{iqk} \quad \forall (i, j) \in IS \quad (33)$$

Block Time Constraints: (34) ensures that block times are identified as the maximum operation time among the operations assigned to the block.

$$x_{iqk} t_i \leq F_{qk} \quad \forall i, q, \quad k = E_i, \dots, L_i \quad (34)$$

Sign Constraints: (35) and (36) specify the binary requirements of the decision variables. y_{qk} variables are used only in the models that explicitly include them in their objective functions.

$$x_{iqk} = \{0, 1\} \quad \forall i, q, \quad k = E_i, \dots, L_i \quad (35)$$

$$y_{qk} = \{0, 1\} \quad \forall q, k \quad (36)$$

We next introduce our block formation models, referred to as Model B0, Model B1, Model B2 and Model B3.

Model B0 aims solely to find a feasible solution and is expressed as

$$\begin{aligned} & \text{Minimize } 0 \\ & \text{subject to (28), (29a), (30), (31), (32), (33), (35)} \end{aligned}$$

Model B1 aims to find a feasible solution with maximum number of blocks and is expressed as

$$\begin{aligned} & \text{Maximize } \sum_{k=1}^m \sum_{q=1}^b y_{qk} \\ & \text{subject to (28), (29b), (30), (31), (32), (33), (35), (36)} \end{aligned}$$

This model generates many blocks, relaxing the subsequent workstation assignment stage.

Model B2 aims to find a feasible solution with minimum number of blocks and is expressed as

$$\begin{aligned} & \text{Minimize } \sum_{k=1}^m \sum_{q=1}^b y_{qk} \\ & \text{subject to (28), (29b), (30), (31), (32), (33), (35), (36)} \end{aligned}$$

The aim is to obtain a solution with fewer blocks so that the problem size in the workstation assignment stage is reduced.

Model B3 aims to find a feasible solution with minimum total block time and is expressed as

$$\begin{aligned} & \text{Minimize } \sum_{k=1}^m \sum_{q=1}^b F_{qk} \\ & \text{subject to (28), (29a), (30), (31), (32), (33), (34), (35)} \end{aligned}$$

This objective tends to group operations with similar processing times, which aligns with the original cycle time minimization goal.

It is worth noting that although the block formation models are mixed-integer linear programs, their computational complexity is significantly lower than that of the original SP model. This is mainly due to the single scenario structure and the use of min-sum objective functions, which are considerably easier to solve than the min-max cycle time objective of the original formulation.

5.2.2 Stage 2 – Block Assignments to Workstations

Following the block formation stage described in Section 5.2.1, the second stage of the matheuristic focuses on assigning the generated blocks to workstations. In this stage, blocks are treated as aggregated units, and the assignment decisions are made while preserving feasibility with respect to precedence relations, workstation inclusion and exclusion constraints, and scenario dependent cycle time considerations. We refer to the resulting block to workstation assignment formulation as Model W.

Model W takes as input the blocks obtained from the first stage model and determines a feasible assignment of these blocks to workstations. To this end, operation-level information is aggregated at the block level, and updated parameters and decision variables are derived accordingly.

To account for workstation inclusion constraints at the block level, we define block inclusion sets, denoted by IBS_r , in a manner analogous to the operation-level inclusion sets IS_r . Each set IBS_r represent a subset of blocks that must be assigned to the same workstation. The sets IBS_r are constructed using the same

procedure as the operation inclusion sets IS_r . The number of IBS_r sets is less than or equal to the number of blocks.

Example: $n = 15$ and $IS = \{(10,11), (11,12), (13,14)\}$. Then the inclusion subsets, IS_r 's, are $IS_1 = \{1\}, IS_2 = \{2\} \dots IS_9 = \{9\}, IS_{10} = \{10, 11, 12\}, IS_{11} = \{13, 14\}, IS_{12} = \{15\}$. Assume, after block formation model, number of total opened block number is 9, $n_b=9$. Assume, operations 1 to 9 is assigned first 6 blocks and operation 10, 11 is assigned block 7 and operation 12 is assigned block 8. Assume, operation 13, 14, 15 is assigned block 9. Then the inclusion subsets, IBS_r 's, are $IBS_1 = \{1\}, IBS_2 = \{2\} \dots IBS_6 = \{6\}, IBS_7 = \{7, 8\}, IBS_8 = \{9\}$ and number of block-set is equal to 8, R . In addition, if block times of block 7 and 8 are 15,20, respectively, then time of block-set IBS_7 is equal to 35.

The processing time of a block q is defined as the maximum of the operation time among the operations assigned to that block. The processing time of a block inclusion set IBS_r is then computed as the sum of processing times of the blocks assigned to it. In addition, a parameter representing the number of blocks assigned to each block inclusion set is introduced to enforce the limitation on the maximum number of blocks per workstation.

Parameters:

R = number of block-sets (number of defined IBS_r sets) $r = 1, \dots, R$

m_o = max number of blocks per workstation

nb_r = number of blocks in block-set $r, r = 1, \dots, R$

F_r = operation time of block-set $r, r = 1, \dots, R$

m = number of workstations

S = number of scenarios

u_{ks} = maintenance time of workstation k in scenario $s, s = 1, \dots, S, k = 1, \dots, m$

p_s = realization probability of scenario $s, s = 1, \dots, S$

\overline{ES} = set of block-set pairs that cannot be assigned to the same workstation

\overline{ES} represents the workstation exclusion constraints at the block-set level and is derived from the operation level exclusion set ES . Specifically, if for any $(i, j) \in ES$, $i \in \text{block-set } r_1$ and $j \in \text{block-set } r_2$ then $(r_1, r_2) \in \overline{ES}$.

Example: $n = 5$, $R = 8$ and $ES = \{(1,2), (2,3), (4,5)\}$. Assume, operation 1 is assigned block-set r_1 and operation 2 is assigned block-set r_2 . Then, $\overline{ES} = \{1,2\}$.

\overline{E}_r = the earliest workstation that block-set r can be assigned, $r = 1, \dots, R$

\overline{E}_r is defined as the maximum of the earliest allowable workstation indices of the operations assigned to block-set r . This ensures that all operations within the block-set can be feasibly processed at the assigned workstation.

Example: Consider block-set r_1 consisting of operations 1 and 2, where the earliest allowable workstations are $E_1 = 3$ and $E_2 = 2$. Then, the earliest workstation to which block-set r_1 can be assigned is $\overline{E}_1 = \text{Max}\{3, 2\} = 3$.

\overline{L}_r = the latest workstation that block-set r can be assigned, $r = 1, \dots, R$

\overline{L}_r is defined as the minimum of the latest allowable workstation indices of the operations assigned to block-set r . This definition guarantees that the assignment of block-set r to a workstation does not violate the individual feasibility limits of its constituent operations.

Example: Consider block-set r_2 consisting of operations 3 and 4, where the latest allowable workstations are $L_3 = 3$ and $L_4 = 5$. Then, the latest workstation to which block-set r_2 can be assigned is $\overline{L}_2 = \text{Min}\{3, 5\} = 3$.

\overline{P}_r = updated set of all predecessors of block-set r , $r = 1, \dots, R$

\overline{P}_r is obtained by aggregating the operation level immediate precedence relations at the block-set level. Specifically, if an operation i has immediate predecessors set, IP_i then any block-set containing an operation in IP_i is included in \overline{P}_r .

Example: Consider block-set r_3 consisting of operations 5 and 6 such that $IP_5 = \{1, 2, 4\}$ and $IP_6 = \{2, 3\}$. If operations 1, 2 are assigned to block-set r_1 and

block-set r_2 , respectively, then the set of predecessor block-sets of block-set r_3 is $\overline{P}_3 = \{1, 2\}$.

\overline{S}_r = updated set of all successors of block-set $r, r = 1, \dots, R$

\overline{S}_r is defined analogously using operation level immediate successor relations. In particular, if operation i has set of immediate successors, IS , and $i \in$ block – set r then any block-set containing an operation in IS_i is included in \overline{S}_r .

Example: Consider block-set r_3 consisting of operations 2 and 5 such that $IS_2 = \{6\}$ and $IS_5 = \{7\}$. If operations 6 and 7 are assigned to block-set r_4 and block-set r_5 , respectively, then the set of successor block-sets of block-set r_3 is $\overline{S}_3 = \{4, 5\}$.

Decision variables:

$$y_{rk} = \begin{cases} 1, & \text{block – set } r \text{ is assigned to workstation } k \\ 0, & \text{otherwise} \end{cases} \quad \forall r, k$$

$$CT_s = \text{cycle time in scenario } s \quad \forall s$$

Constraints:

Assignment Constraints: (37) ensures that each block-set is assigned to exactly one workstation.

$$\sum_{k \in \overline{E}_r} y_{rk} = 1 \quad \forall r \quad (37)$$

Workstation Opening Constraints: (38) links block-set assignments to workstation opening decisions and limits the number of blocks assigned to an opened workstation.

$$\sum_{r=1}^R n_b y_{rk} \leq m_0 \quad \forall k \quad (38)$$

Precedence Constraints: (39) enforces block level precedence relations by ensuring that each block-set is assigned to a workstation that does not violate the precedence structure.

$$\sum_{h=\overline{E}_g}^{\overline{L}_g} hy_{gh} \leq \sum_{s=\overline{E}_r}^{\overline{L}_r} sy_{rs} \quad \forall r, \forall g \in \overline{P}_r \quad (39)$$

Workstation Exclusion Constraints: (40) prevents block-set pairs with exclusion requirements from being assigned to the same workstation.

$$y_{rk} \leq (1 - y_{gk}) \quad \forall (r, g) \in \overline{ES}, k = \overline{E}_r, \dots, \overline{L}_r \quad (40)$$

Same Workstation Constraints: (41) ensures that block-set pairs that have workstation inclusion constraints are assigned to the same workstation.

$$\sum_{k=\overline{E}_r}^{\overline{L}_r} ky_{rk} = \sum_{k=\overline{E}_g}^{\overline{L}_g} ky_{gk} \quad \forall (r, g) \in \overline{IS} \quad (41)$$

Cycle Time Constraints: (42) defines the cycle time for each scenario as the maximum workload across all workstations, including both block-set processing times and scenario-dependent maintenance times.

$$u_{ks}z_k + \sum_{r=1}^R y_{rk}F_r \leq CT_s \quad \forall s, k \quad (42)$$

Sign Constraints: (43) and (44) represent the binary requirements.

$$y_{rk} = \{0, 1\} \quad \forall r, k = \overline{E}_r, \dots, \overline{L}_r \quad (43)$$

$$z_k = \{0, 1\} \quad \forall k \quad (44)$$

The objective function is to minimize the expected cycle time across all scenarios, which is formulated as.

$$\text{Min } \sum_{s=1}^S p_s CT_s \quad (45)$$

Model W can be viewed as a multi scenario assembly line balancing problem with workstation inclusion and exclusion constraints that are analogous to zoning marriage and zoning divorce constraints. Building on this structure, it is worth noting that although Model W is formulated as a mixed-integer linear program and retains a min-max cycle time objective, its computational complexity is significantly lower than that of the original SP model.

This reduction in complexity primarily stems from the aggregation of operations into blocks and the use of block inclusion sets, which substantially decrease the number of decision variables and constraints. In addition, the sequential decomposition of the solution process further improves tractability. By fixing the block structure in advance, Model W avoids the simultaneous operation to block and block to workstation assignment decisions present in the SP model, resulting in a more manageable formulation for large-sized instances.

5.2.3 Improvement Procedures

In this section, we introduce two improvement procedures designed to enhance a given feasible solution by reducing the cycle time. Both procedures focus on workload imbalances across workstations, with particular emphasis on the most heavily loaded workstation, which determines the current cycle time. While the first procedure relies on exchange based local improvements, the second employs a restricted model-based optimization strategy.

5.2.3.1 Improvement Procedure 1

Improvement Procedure 1 directly addresses workload imbalance by focusing on the most heavily loaded workstation. Accordingly, the procedure first explores block exchanges between the most heavily loaded workstation and the remaining ones and selects the exchange that maximizes improvement in cycle time value.

To limit the scope of the search, we allow at most one exchange between two specified workstations. Subsequently, we conduct operation exchanges within the blocks of the most loaded workstation and those of others, selecting two operations that yield the maximum improvement. As in the block exchange phase, at most one operation exchange is allowed between a given pair.

The process terminates once all feasible operation pairs between the current most loaded workstation and others have been considered. Block and operation exchanges cease once there is no improvement in cycle time value.

Below is the algorithmic description of Improvement Procedure 1.

Algorithm:

Step 1: Initial Evaluation

Given a feasible solution, compute the workload of each workstation.

Identify the most heavily loaded workstation, i.e., the workstation that determines the current cycle time.

Step 2: Block Exchange Phase

For the most loaded workstation, identify all blocks assigned to it. Then, for each other workstation:

- Consider all feasible block exchange pairs between the most loaded workstation and the selected workstation.
- Evaluate the resulting cycle time for each feasible block exchange.

Among all evaluated block exchanges:

- Select the exchange that yields the maximum reduction in cycle time.

If an improving block exchange exists:

- Apply the selected exchange.
- Update the solution and recompute workstation workloads.

Repeat step 2 until all feasible block pairs between the current most loaded workstation and the remaining workstations have been examined. Once no further improving block exchange is available, the block exchange phase is terminated.

Step 3: Operation Exchange Phase

Identify the current most heavily loaded workstation.

For each block of the most loaded workstation and each block of the remaining workstations:

- Consider all feasible operation exchange pairs.
- Evaluate the resulting cycle time.

Among all evaluated operation exchanges:

- Select the operation pair that produces the maximum improvement in cycle time.

If an improving operation exchange exists:

- Apply the selected exchange.
- Update the solution and recompute workstation workloads.

Repeat step 3 until all feasible operation pairs have been examined and no operation exchange yields an improvement in cycle time.

The final solution obtained at the end of these steps is returned as the improved solution generated by Improvement Procedure 1.

5.2.3.2 Improvement Procedure 2

Improvement Procedure 2 adopts an iterative, model-based re-optimization strategy to refine a given feasible solution. At each iteration, the procedure focuses on the most heavily loaded (bottleneck) workstation, which determines the current cycle time, and improves the solution through a sequence of restricted optimization problems defined over workstation pairs.

In each iteration, the bottleneck workstation is paired, one at a time, with each of the remaining workstations. For a selected pair, all block and operation assignments on these two workstations are released, while the assignments of all other workstations are fixed to their values in the current best solution. A restricted

version of the original SP formulation is then solved to optimally reassign blocks and operations between the selected workstation pair.

This pairwise re-optimization process is carried out for all $(m - 1)$ pairs involving the bottleneck workstation in a system with (m) workstations. Among the solutions obtained within an iteration, the one with the minimum expected cycle time is retained as the updated best solution. The overall procedure is repeated until no further improvement is observed or a preset iteration limit of m is reached. Consequently, at most $m(m - 1)$ restricted models, each involving only two workstations, are solved throughout the procedure.

In each restricted model, decision variables are defined only for the bottleneck workstation and the paired workstation under consideration. All assignment variables corresponding to the remaining workstations are fixed to their values in the current solution.

To formalize this restriction, the parameter

$$y_{iqk} = \begin{cases} 1, & \text{operation } i \text{ is assigned to block } q \text{ in workstation} \\ & k \text{ in the current best solution} \\ 0, & \text{otherwise} \end{cases}$$

is introduced and used to fix the corresponding assignment variables. As a result, the restricted model includes decision variables x_{iqk} only for the selected workstation pair, leading to a significantly reduced problem involving a limited set of operations. This reduction enables an exact solver to efficiently explore a richer neighborhood than simple exchange-based moves.

By iteratively allowing simultaneous reassignment decisions at both the block and operation levels for workstation pairs, Improvement Procedure 2 provides stronger improvement capability than Improvement Procedure 1, however at a higher computational cost.

Below is the algorithmic description of Improvement Procedure 2:

Algorithm:

Step 1: Initial Evaluation

Given a feasible solution, compute the workload of each workstation.
Identify the most heavily loaded (bottleneck) workstation

Step 2: Workstation Pairing

Pair the bottleneck workstation with each of the remaining workstations,
one at a time.

Step 3: Pairwise Optimization Phase

For the selected workstation pair:

- Release all operation and block assignments on the two workstations.
- Fix the assignments of all other workstations.
- Solve the restricted SP model involving only the selected pair.

Step 4: Solution Update

Evaluate the resulting solution and retain it if it improves the best solution found so far.

Step 5: Iteration Control

Repeat Steps 2–4 for all workstation pairs involving the bottleneck workstation.

Repeat the entire procedure until no further improvement is obtained or the maximum number of iterations (m) is reached.

The best solution obtained is returned as the final solution produced by Improvement Procedure 2.

5.3 Rule-based Heuristic

The Rule-based heuristic is designed to rapidly generate a feasible initial solution in polynomial time. It follows a deterministic, stepwise structure based on simple yet effective assignment rules, and serves as a foundation for the subsequent improvement procedures. The heuristic uses the following three steps:

Step 1. Block Formation Rule

Step 2. Block Assignments Rule

Step 3. Improvement Procedure 1

5.3.1 Block Formation Rule

We start by sorting the operations in non-increasing order of operation times. Operations are then assigned to blocks while adhering to precedence, block and workstation exclusion constraints, and operation per block limitation. If no further operations can be assigned to the current block without violating these limitations, we proceed to the next block. This process continues until all operations are assigned to blocks.

In the first step, operations are grouped into blocks and blocks are grouped into block-sets by following a set of precedence and feasibility preserving rules. The goal is to form compact blocks while respecting structural constraints and limiting the number of operations per block.

Algorithm:

Step 1: Sort the operations in non-increasing order of their processing times.

Step 2: Initialize the block index b and the set of unassigned operations U .

Step 3: While $U \neq \emptyset$, perform the following steps:

- Initialize the current block $b \leftarrow \emptyset$
- For each operation $i \in U$ (in sorted order), check whether assigning operation i to the current block violates any of the following constraint

- precedence constraints
- block and workstation exclusion constraints
- operation per block limitation
- If no violation occurs, assign operation i to block b and remove it from U

If no further operation in U can be assigned to block b , increment the block index and initiate a new block.

Step 4: Repeat until all operations are assigned to blocks.

Step 5: Construct the block-sets, IBS_r and find the processing times of the block-sets and the number of blocks in each block-set.

Due to workstation inclusion constraints and the limitation on the maximum number of blocks per workstation, infeasibility may arise during the subsequent block assignment stage. To avoid unnecessary computation, infeasibility is checked immediately after block formation using the following conditions:

- $|IBS_r| > m_0$ for any r
- Any IBS_r contains two operations whose corresponding operations are subject to workstation exclusion constraints.

If infeasibility is detected, the heuristic is terminated in Step 1.

5.3.2 Block Assignments Rule

After verifying feasibility, the second stage assigns the generated block-sets, IBS_r , to workstations. At this stage, blocks are treated as aggregated operations. Starting from the first workstation, eligible block-sets are sequentially assigned to workstations while preserving feasibility with respect to precedence relations, workstation exclusion constraints, and the maximum number of blocks allowed per workstation. If no further block-set can be assigned to the current workstation, the procedure proceeds to the next workstation. The process continues until all block-

sets are assigned to workstations or the maximum number of workstations is reached.

Algorithm:

Step 1: Sort the block-sets in non-increasing order of their processing times.

Step 2: Initialize the workstation index k and the set of unassigned block-sets U .

Step 3: While $U \neq \emptyset$, perform the following steps:

- Initialize the current workstation $k \leftarrow 1$
- For each block-set $r \in U$ (in sorted order), check whether assigning block-set r to the current workstation violates any of the following constraints
 - precedence constraints
 - workstation exclusion constraints
 - block per workstation limitation
- If no violation occurs, assign block-set r to workstation k and remove it from U ,
- If no further block-set in U can be assigned to the current workstation, increment the workstation index.

Step 4: Repeat until all block-sets are assigned to workstation or workstation limit is reached.

5.3.3 Improvement Procedure 1

The initial solution obtained from the block formation and block assignment rules may still exhibit workload imbalances across workstations. To address this issue, Improvement Procedure 1 is applied as a final stage of the rule-based heuristic. This procedure iteratively evaluates block and operation exchanges between the bottleneck workstation and the remaining workstations, and applies the exchange that yields the largest reduction in the expected cycle time value. The procedure terminates when no further block or operation exchange leads to an improvement.

5.3.4 Computational Complexity

The Rule-based heuristic is polynomially bounded and can be executed efficiently even for large-sized problem instances. The block formation and block assignment steps rely on sorting operations and blocks, followed by sequential feasibility checks, each of which can be performed in polynomial time with respect to the number of operations and blocks. Improvement Procedure 1 explores a restricted local neighborhood by considering block and operation exchanges involving only the bottleneck workstation, which further limits the computational burden.

The Numeric Example (cont.)

Recall the problem using a sample instance taken from Belmokhtar et al. (2006), which addresses the machining line design for the manufacturing of a mechanical desktop. First, we solve the block formation models B0, B1, B2, and B3 and obtain the following assignments.

Table 21: Assignments of operations to blocks Model B0

| Block | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|------------|-----|-----|----|----|--------|------|----|----|----|
| Operations | 3,6 | 4,7 | 5 | 8 | 1,9,12 | 2,10 | 11 | 13 | 14 |
| Block Time | 14 | 19 | 13 | 17 | 17 | 16 | 20 | 13 | 17 |

Table 22: Assignments of operations to blocks Model B1

| Block | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|------------|-----|-----|----|----|----------|----|----|----|----|
| Operations | 3,6 | 4,7 | 5 | 8 | 1,2,9,12 | 10 | 11 | 13 | 14 |
| Block Time | 14 | 19 | 13 | 17 | 17 | 16 | 20 | 13 | 17 |

Table 23: Assignments of operations to blocks Model B2

| Block | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------------|-----|-------|-----------|-----|----|----|----|
| Operations | 3,6 | 1,2,9 | 4,7,10,12 | 5,8 | 11 | 13 | 14 |
| Block Time | 14 | 17 | 19 | 17 | 20 | 13 | 17 |

Table 24: Assignments of operations to blocks Model B3

| Block | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------------|---------|----------|-----|----|----|----|----|
| Operations | 2,3,6,9 | 1,4,7,10 | 5,8 | 11 | 12 | 13 | 14 |
| Block Time | 17 | 19 | 17 | 20 | 10 | 13 | 17 |

Next, Model W is solved.

Table 25: Assignments of blocks to workstations Model B0-W

| Workstation | 1 | | 2 | | 3 | | 4 | 5 | |
|-------------|------|------|----|----|----------|-------|----|----|----|
| Block | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 |
| Block-sets | 1 | 2 | 1 | 2 | 1 | | 1 | 1 | 2 |
| Operations | 3, 6 | 4, 7 | 5 | 8 | 1, 9, 12 | 2, 10 | 11 | 13 | 14 |
| Block Time | 14 | 19 | 13 | 17 | 17 | 16 | 20 | 13 | 17 |

With objective value 41.8125.

Table 26: Assignments of blocks to workstations Model B1-W

| Workstation | 1 | | 2 | | 3 | | 4 | 5 | |
|-------------|------|------|----|----|-------------|----|----|----|----|
| Block | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 |
| Block-sets | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 |
| Operations | 3, 6 | 4, 7 | 5 | 8 | 1, 2, 9, 12 | 10 | 11 | 13 | 14 |
| Block Time | 14 | 19 | 13 | 17 | 17 | 16 | 20 | 13 | 17 |

With objective value 41.8125.

Table 27: Assignments of blocks to workstations Model B2-W

| Workstation | 1 | | 2 | 3 | 4 | 5 | |
|-------------|------|---------|--------------|------|----|----|----|
| Block | 1 | 2 | 1 | 1 | 1 | 1 | 2 |
| Block-sets | 1 | 2 | 1 | 1 | 1 | 1 | 2 |
| Operations | 3, 6 | 1, 2, 9 | 4, 7, 10, 12 | 5, 8 | 11 | 13 | 14 |
| Block Time | 14 | 17 | 19 | 17 | 20 | 13 | 17 |

With objective value 38.25.

Table 28: Assignments of blocks to workstations Model B3-W

| Workstation | 1 | | 2 | 3 | 4 | | 5 |
|-------------|------------|-------------|------|----|----|----|----|
| Block | 1 | 2 | 1 | 1 | 1 | 2 | 1 |
| Block-sets | 1 | | 1 | 1 | 1 | 2 | 1 |
| Operations | 2, 3, 6, 9 | 1, 4, 7, 10 | 5, 8 | 11 | 12 | 13 | 14 |
| Block Time | 17 | 19 | 17 | 20 | 10 | 13 | 17 |

With objective value 41.

After finding a feasible solution, improvement procedures are applied, separately.

Table 29: Improved solution of Model B0-W-IP1

| Workstation | 1 | | 2 | 3 | 4 | 5 | |
|-------------|----------|---------|------|----------|----|----|----|
| Block-sets | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Operations | 3, 6, 12 | 4, 7, 9 | 5, 8 | 1, 2, 10 | 11 | 13 | 14 |
| Block Time | 14 | 19 | 17 | 16 | 20 | 13 | 17 |

With objective value 39.75.

Table 30: Improved solution of Model B1-W-IP1

| | | | | | | | |
|--------------------|----------|---------|------|----------|----|----|----|
| Workstation | 1 | 1 | 2 | 3 | 4 | 5 | 5 |
| Block-sets | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Operations | 3, 6, 12 | 4, 7, 9 | 5, 8 | 1, 2, 10 | 11 | 13 | 14 |
| Block Time | 14 | 19 | 17 | 16 | 20 | 13 | 17 |

With objective value 39.75.

Table 31: Improved solution of Model B2-W-IP1

| | | | | | | | |
|--------------------|------|---------|--------------|------|----|----|----|
| Workstation | 1 | | 2 | 3 | 4 | 5 | |
| Block-sets | 1 | 2 | 1 | 1 | 1 | 1 | 2 |
| Operations | 3, 6 | 1, 2, 9 | 4, 7, 10, 12 | 5, 8 | 11 | 13 | 14 |
| Block Time | 14 | 17 | 19 | 17 | 20 | 13 | 17 |

With objective value 38.25.

Table 32: Improved solution of Model B3-W-IP1

| | | | | | | | |
|--------------------|------------|-------------|------|----|----|----|----|
| Workstation | 1 | | 2 | 3 | 4 | | 5 |
| Block-sets | 1 | | 2 | 3 | 4 | 5 | 6 |
| Operations | 2, 3, 6, 9 | 1, 4, 7, 10 | 5, 8 | 11 | 12 | 13 | 14 |
| Block Time | 17 | 19 | 17 | 20 | 10 | 13 | 17 |

With objective value 41.

Table 33: Improved solution of Model B0-W-IP2

| | | | | | | | |
|--------------------|----------|---------|----------|------|----|----|----|
| Workstation | 1 | | 2 | 3 | 4 | 5 | |
| Block-sets | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Operations | 3, 6, 12 | 4, 7, 9 | 1, 2, 10 | 5, 8 | 11 | 13 | 14 |
| Block Time | 14 | 19 | 16 | 17 | 20 | 13 | 17 |

With objective value 39.75.

Table 34: Improved solution of Model B1-W-IP2

| | | | | | | | |
|--------------------|----------|---------|----------|------|----|----|----|
| Workstation | 1 | | 2 | 3 | 4 | 5 | |
| Block-sets | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Operations | 3, 6, 12 | 4, 7, 9 | 1, 2, 10 | 5, 8 | 11 | 13 | 14 |
| Block Time | 14 | 19 | 16 | 17 | 20 | 13 | 17 |

With objective value 39.75.

Table 35: Improved solution of Model B2-W-IP2

| | | | | | | | |
|--------------------|------|---------|--------------|------|----|----|----|
| Workstation | 1 | | 2 | 3 | 4 | 5 | |
| Block-sets | 1 | 2 | 1 | 1 | 1 | 1 | 2 |
| Operations | 3, 6 | 1, 2, 9 | 4, 7, 10, 12 | 5, 8 | 11 | 13 | 14 |
| Block Time | 14 | 17 | 19 | 17 | 20 | 13 | 17 |

With objective value 38.25.

Table 36: Improved solution of Model B3-W-IP2

| Workstation | 1 | | 2 | 3 | 4 | 5 | |
|-------------|----------|---------|----------|------|----|----|----|
| Block-sets | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Operations | 3, 6, 12 | 4, 7, 9 | 1, 2, 10 | 5, 8 | 11 | 13 | 14 |
| Block Time | 14 | 19 | 16 | 17 | 20 | 13 | 17 |

With objective value 39.75.

For Rule-based heuristic, we solved Block Formation Rule (BFR) and Block Assignment Rule (BAR).

Table 37: Assignments of operations to blocks Model BFR

| Block | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|------------|------|------|----|----|-------------|----|----|----|----|
| Operations | 3, 6 | 4, 7 | 5 | 8 | 1, 2, 9, 12 | 10 | 11 | 13 | 14 |
| Block Time | 14 | 19 | 13 | 17 | 17 | 16 | 20 | 13 | 17 |

Table 38: Assignments of blocks to workstations Model BFR-BAR

| Workstation | 1 | | 2 | | 3 | | 4 | 5 | |
|-------------|------|------|----|----|-------------|----|----|----|----|
| Block | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 |
| Block-sets | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 |
| Operations | 3, 6 | 4, 7 | 5 | 8 | 1, 2, 9, 12 | 10 | 11 | 13 | 14 |
| Block Time | 14 | 19 | 13 | 17 | 17 | 16 | 20 | 13 | 17 |

With objective value 41.25.

After finding a feasible solution, improvement procedure 1 is applied.

Table 39: Improved solution of Model BFR-BAR-IP1

| Workstation | 1 | | 2 | 3 | 4 | 5 | |
|-------------|----------|---------|------|----------|----|----|----|
| Block-sets | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Operations | 3, 6, 12 | 4, 7, 9 | 5, 8 | 1, 2, 10 | 11 | 13 | 14 |
| Block Time | 14 | 19 | 17 | 16 | 20 | 13 | 17 |

With objective value 40.5.

Table 40: Final results of Model- and Rule-based heuristics

| Model | Number of Blocks | Number of Block-sets | Model W Objective Value | Model BAR Objective Value | Model IP1 Objective Value | Model IP2 Objective Value |
|-----------------|------------------|----------------------|-------------------------|---------------------------|---------------------------|---------------------------|
| Model B0 | 9 | 8 | 41.8125 | - | 39.7500 | 39.7500 |
| Model B1 | 9 | 9 | 41.8125 | - | 39.7500 | 39.7500 |
| Model B2 | 7 | 7 | 38.2500 | - | 38.2500 | 38.2500 |
| Model B3 | 7 | 6 | 41.0000 | - | 41.0000 | 39.7500 |
| Model BFR | 9 | 9 | - | 41.2500 | 40.5000 | - |

To sum up, for this specific example, the optimal value is 38.25 and Model B2-based metaheuristic provides the optimal solution and Rule-based heuristic objective value is 40.5 as presented in Table 40.

CHAPTER 6

COMPUTATIONAL EXPERIMENT

In this section, we evaluate the performance of the proposed solution approaches. First, the data generation scheme and the performance measures are described. Then, the results obtained from the preliminary and main experiments are presented.

The models are implemented using C++ (CLion version 2024.3.3) and solved by CPLEX 22 software. All experiments were conducted on a computer with an Intel(R) Core (TM) i5 – 6600K CPU @3.50GHz and 8.00 GB RAM.

6.1 Data Generation

Guschinskaya and Dolgui (2006) provided a set of benchmark problems with 25, 50 and 100 operations and 4, 5, 10, 15 workstations where the operation times are drawn from a uniform distribution $U[10,20]$. Telemeci and Azizoğlu (2022) modified these benchmark networks by adjusting the number of immediate predecessor relations and incorporating block and workstation exclusion constraints. In this study, the dataset is further extended by including workstation inclusion constraints, and the resulting instances are utilized in the computational experiments.

Based on the structural characteristics of real-world transfer line systems and commonly adopted assumptions in the related literature, the following parameter settings are used in the main experiments:

- **Number of operations: 15, 20, 25, 30, 40, 50, 75, 100**

Those values represent both small and large transfer line configurations. Smaller instances correspond to relatively simple machining lines and allow the validation of exact solution approaches, whereas larger instances model complex industrial transfer lines with a high degree of operation divisions, where precedence, block, and workstation-related constraints are more common.

- **Number of workstations: 4 and 6**

Those values reflect compact transfer line layouts frequently observed in practice. Such limitations are typically driven by investment costs and floor space constraints. Restricting the number of workstations increases the operational load per workstation and intensifies the interaction between assignment, block formation, and exclusion constraints, thereby capturing realistic congestion effects in transfer line systems.

- **Number of operations per block and blocks per workstation: 4**

Both the maximum number of operations per block and the maximum number of blocks per workstation are set to four. In transfer line balancing, blocks represent groups of operations that share common tooling or fixture configurations. Physical limitations on machining heads and tooling magazines restrict the number of tools that can be installed simultaneously, making these bounds consistent with practical transfer line implementations.

- **Operation time distributions $U[1, 20]$ and $U[10, 20]$**

The distribution $U[10, 20]$ reflects relatively balanced machining environments in which operations are engineered to have comparable durations, a common assumption in transfer line design. This is also consistent with standard literature to ensure that solution difficulty is driven by structural constraints rather than extreme processing time outliers. In contrast, the distribution $U[1, 20]$ introduces higher variability and represents less balanced or mixed-production settings,

allowing the evaluation of algorithmic robustness under increased processing time heterogeneity.

- **Maintenance Case Types: Case 1 and Case 2**

To examine how different maintenance configurations affect solution difficulty and to reflect realistic operating conditions in transfer line systems, two distinct maintenance cases are considered. These cases are designed to represent varying levels of disruption commonly observed in machining transfer lines, ranging from uninterrupted operation to moderate and severe breakdown events.

Case 1 which is used in the main experiments, consists of two possible maintenance time realizations defined as

$$u_{ks} = \begin{cases} 0 & \text{with } \pi_{k1} \text{ probability} \\ \sum t_i/N & \text{with } \pi_{k2} \text{ probability} \end{cases} \quad \forall k, s$$

where π_{k1} and π_{k2} are the realization probability of maintenance time type 1 and type 2, respectively.

The value, $u_{ks} = 0$, represents situations in which no corrective maintenance is required, such as normal operation, preventive maintenance performed outside production hours, or very minor disturbances that do not interrupt the production flow. The second maintenance type, $\sum t_i/N$, corresponds to moderate corrective maintenance activities whose duration is comparable to the average processing time of an operation, such as tool replacement or minor mechanical adjustments.

For the identical probability setup, both maintenance types are assumed to be equally likely, i.e., therefore, π_{k1} and π_{k2} are equal to $1/2$. For the non-identical probability setup, minor or no-maintenance events are assumed to occur more frequently, and the probabilities are set to, $\pi_{k1} = 3/4$ and $\pi_{k2} = 1/4$.

Case 2 extends Case 1 by introducing a third maintenance type in order to capture more disruptive operating conditions:

$$u_{ks} = \begin{cases} 0 & \text{with } \pi_{k1} \text{ probability} \\ \sum t_i/N & \text{with } \pi_{k2} \text{ probability} \\ \sum t_i & \text{with } \pi_{k3} \text{ probability} \end{cases} \quad \forall k, s$$

where π_{k1} , π_{k2} and π_{k3} are the realization probability of maintenance time type 1, type 2 and type 3, respectively.

In this case, the first two maintenance types have the same interpretation as in Case 1. The third maintenance type, $\sum t_i$ represents severe breakdowns or shutdown conditions, such as major mechanical failures or system-wide malfunctions, leading to prolonged downtime. This type models rare but high-impact events that can significantly disrupt production in transfer line systems.

For the identical probability setup, all three maintenance types are assumed to be equally likely, i.e., $\pi_{k1} = \pi_{k2} = \pi_{k3} = 1/3$. For the non-identical probability setup, π_{k1} , π_{k2} and π_{k3} are set to $2/4$, $1/4$ and $1/4$, respectively, reflecting operating environments in which minor disruptions are more common than severe breakdowns.

By distinguishing between these maintenance cases and probability structures, the experimental design allows for a systematic evaluation of algorithmic performance under both stable operating conditions and highly disruptive scenarios representative of real transfer line systems.

- **Precedence Setup Structure: Loose and Tight**

To analyze how constraint density affects solution difficulty, we generated two distinct precedence structures and this differentiation allows us to observe if the algorithms perform differently under flexible (loose) versus restricted (tight) conditions.

The total number of possible precedence-related relations is proportional to $n(n - 1)/2$ where n denotes the number of operations. Based on this measure, two sets are defined.

Set Loose (L) – The loose configuration represents transfer line systems with relatively flexible process planning and fewer technological or physical restrictions. In this setting,

$$\text{Number of Immediate Predecessors (IP)} = 0.05 * n(n - 1) / 2$$

$$\text{Number of Block Exclusion Relations (EB)} = 0.02 * n(n - 1) / 2$$

$$\text{Number of Workstation Exclusion Relations (ES)} = 0.02 * n(n - 1) / 2$$

$$\text{Number of Workstation Inclusion Relations (IS)} = 0.01 * n(n - 1) / 2$$

This structure models production environments where alternative workstation assignments are largely permissible.

Set Tight (T) – The tight configuration models highly constrained transfer line systems, where technological requirements, tooling limitations, and safety considerations impose stricter precedence and assignment rules. To create tight set, the numbers of block exclusion, workstation exclusion, and workstation inclusion relations in the Loose Set are doubled. Accordingly,

$$\text{Number of Immediate Predecessors (IP)} = 0.10 * n(n - 1) / 2$$

$$\text{Number of Block Exclusion Relations (EB)} = 0.04 * n(n - 1) / 2$$

$$\text{Number of Workstation Exclusion Relations (ES)} = 0.04 * n(n - 1) / 2$$

$$\text{Number of Workstation Inclusion Relations (IS)} = 0.02 * n(n - 1) / 2$$

Such a setup reflects rigid production environments with limited assignment flexibility.

6.2 Performance Measures

To evaluate the effectiveness and efficiency of the proposed solution methods, we employ specific performance metrics tailored to each algorithmic approach. For every experimental setup, we generate and solve ten problem instances.

6.2.1 Mathematical Models

For the exact mathematical models—Stochastic Programming (SP) and Constraint Programming (CP)—computational efficiency is the primary concern given their complexity. We evaluate their performance using the following time-based metrics:

- Average Central Processing Unit (CPU) Time: The mean of the computation time required to solve the ten instances in each dataset.
- Maximum Central Processing Unit (CPU) Time: The longest computation time observed among the ten instances, indicating the worst-case performance.

A termination limit of 7200 seconds (2 hours) is imposed on the execution of both mathematical models.

6.2.2 Branch and Bound (B&B) Algorithm

The performance of the B&B algorithm is evaluated using a comprehensive set of metrics that capture both computational speed and search tree characteristics:

- Average & Maximum CPU Time: The mean and the longest of the computation time required to solve the ten instances in each dataset.
- Solution Quality: The average and maximum percentage deviation of the best-found solution from the optimal solution or the best-known upper bound.

- **Time to Best Solution:** The CPU time at which the algorithm identifies the optimal or best-known solution. This highlights the algorithm's ability to find high-quality solutions early in the search.
- **Total Nodes Explored:** The average and maximum number of nodes generated in the search tree, measuring the magnitude of the search space.

Similar to the mathematical models, a termination limit of 7200 seconds (2 hours) is applied to the Branch and Bound algorithm.

6.2.3 Heuristic Procedures

For the heuristic approaches, the focus shifts to the trade-off between solution quality and computational speed. Distinct time limits are applied based on the nature of the heuristic.

- **B&B-based Heuristic:** A time limit of 7200 seconds (2 hours).
- **Model- and Rule-based Heuristics:** A time limit of 900 seconds (15 minutes).

6.2.3.1 B&B-based Heuristic

The performance of the B&B-based heuristic is evaluated by comparing its computational efficiency with that of the standard B&B algorithm:

- **Average & Maximum CPU Time:** The mean and the longest of the computation time required to solve the ten instances in each dataset.
- **Solution Quality:** The average and maximum percentage deviation of the heuristic solution from the optimal solution or the best known upper bound.

6.2.3.2 Model- and Rule-based Heuristics

The performances of the matheuristics are analyzed through comparative metrics involving the underlying models and other heuristics:

- Average & Maximum CPU Time: The mean and the longest of the computation time required to solve the ten instances in each dataset.
- Solution Quality: The average and maximum percentage deviation of the heuristic solution from the optimal solution or the best-known upper bound.
- Relative to Mathematical Models: We compare the solution quality (Cycle Time) and CPU time against the exact SP and CP models.
- Frequency Best: To evaluate the comparative efficacy of the block formation strategies, we analyze the frequency with which each method (B0, B1, B2, B3) achieves the best solution among the set.
- Improvement Procedure Effectiveness: We quantify the impact of the local search and re-optimization phases by comparing the objective function values before and after the application of improvement procedures.
- Relative to Rule-based heuristic: We calculate the percentage improvement in the objective function achieved by the matheuristics over the baseline Rule-based heuristic. This quantifies the value added by the mathematical improvement phases.
- Relative Comparisons: We compare the effectiveness of the different block formation strategies relative to each other (Model B2 versus Model B3)
- Structural Metrics: The average and maximum number of blocks and number of block-sets generated by the different block formation strategies.

As a constructive method, the Rule-based heuristic serves as the baseline for all comparisons. Its performance is measured by:

- Average & Maximum CPU Time: The mean and the longest of the computation time required to solve the ten instances in each dataset. As this

method is polynomially bounded, we verify that it generates solutions in negligible CPU time.

- Feasibility Frequencies: the number of instances for which a feasible solution is found.
- Infeasibility Detection: The count of instances where infeasibility was correctly detected by the predefined infeasibility conditions.

6.3 Preliminary Experiments

Before conducting the main computational study, a series of preliminary experiments is performed to better understand the behavior of the mathematical models, the B&B algorithm and the heuristic approaches. The objectives of these experiments are to identify the factors that significantly affect computational performance and eliminate unnecessary complexity in the main experiments.

6.3.1 Mathematical Models

We first investigate whether the dispersion of operation processing times influences computational performance. Intuitively, wider distributions (e.g., $U[1,20]$) may create more heterogeneous workload allocations across workstations, potentially increasing combinatorial complexity.

The following setup is used:

- Number of operations: 15
- Number of workstations: 4 and 6
- Number of operations per block: 4
- Number of blocks per workstation: 4
- Maintenance type: Case 2
- Maintenance probabilities: Non-identical probabilities
- Precedence structure: Tight and loose

- Operation time distributions: U[1, 20] and U[10, 20]
- Earliest and latest workstation parameters (E_i and L_i): Included and not-included

The results are reported in Table 41.

Table 41: The effect of distribution of processing times of operations

| m | Setup | E_i&L_i | U[10, 20] | SP Avg. CPU Time | SP Max. CPU Time |
|----------|--------------|--|------------------|-------------------------|-------------------------|
| 4 | T | ✗ | ✗ | 1.08 | 2.56 |
| 4 | T | ✗ | ✓ | 1.13 | 2.58 |
| 4 | L | ✗ | ✗ | 1.35 | 2.92 |
| 4 | L | ✗ | ✓ | 1.34 | 3.00 |
| 4 | T | ✓ | ✗ | 0.86 | 1.83 |
| 4 | T | ✓ | ✓ | 0.81 | 1.63 |
| 4 | L | ✓ | ✗ | 1.33 | 2.97 |
| 4 | L | ✓ | ✓ | 1.30 | 3.02 |
| 6 | T | ✗ | ✗ | 21.76 | 41.03 |
| 6 | T | ✗ | ✓ | 21.89 | 40.56 |
| 6 | L | ✗ | ✗ | 24.05 | 36.86 |
| 6 | L | ✗ | ✓ | 24.20 | 36.73 |
| 6 | T | ✓ | ✗ | 17.72 | 32.88 |
| 6 | T | ✓ | ✓ | 17.65 | 33.02 |
| 6 | L | ✓ | ✗ | 24.02 | 36.22 |
| 6 | L | ✓ | ✓ | 23.96 | 36.61 |

Across all configurations, the difference between U[1, 20] and U[10, 20] is negligible in terms of both average and maximum CPU times for SP model. This indicates that the computational burden is primarily driven by other parameters (number of workstations, precedence tightness, etc.) rather than the dispersion of processing times.

Since no systematic computational impact is observed, the remaining experiments are conducted using U[10, 20] to maintain consistency while avoiding redundant parameter variation.

We next analyze whether incorporating earliest and latest workstation bounds improves performance. From a modeling perspective, E_i and L_i reduce the feasible assignment region by tightening variable domains. Therefore, they are expected to strengthen the formulation and reduce branch-and-bound effort.

The results are presented in Table 42.

Table 42: The effect of earliest and latest workstation information

| m | U[10, 20] | Setup | $E_i \& L_i$ | SP Avg. CPU Time | SP Max. CPU Time |
|----------|------------------|--------------|------------------------------------|-------------------------|-------------------------|
| 4 | ✗ | T | ✗ | 1.08 | 2.56 |
| 4 | ✗ | T | ✓ | 0.86 | 1.83 |
| 4 | ✗ | L | ✗ | 1.35 | 2.92 |
| 4 | ✗ | L | ✓ | 1.33 | 2.97 |
| 4 | ✓ | T | ✗ | 1.13 | 2.58 |
| 4 | ✓ | T | ✓ | 0.81 | 1.63 |
| 4 | ✓ | L | ✗ | 1.34 | 3.00 |
| 4 | ✓ | L | ✓ | 1.30 | 3.02 |
| 6 | ✗ | T | ✗ | 21.76 | 41.03 |
| 6 | ✗ | T | ✓ | 17.72 | 32.88 |
| 6 | ✗ | L | ✗ | 24.05 | 36.86 |
| 6 | ✗ | L | ✓ | 24.02 | 36.22 |
| 6 | ✓ | T | ✗ | 21.89 | 40.56 |
| 6 | ✓ | T | ✓ | 17.65 | 33.02 |
| 6 | ✓ | L | ✗ | 24.20 | 36.73 |
| 6 | ✓ | L | ✓ | 23.96 | 36.61 |

In almost all configurations, especially for $m = 6$, the inclusion of E_i and L_i leads to noticeable reductions in CPU times. The effect is more pronounced in tighter precedence structures and larger workstation settings.

These bound parameters effectively limit the feasible search space, resulting in stronger LP relaxations and fewer explored nodes.

Given their clear computational advantage, E_i and L_i are included in all subsequent experiments.

6.3.2 B&B Algorithm

We next examine the impact of incorporating the proposed lower bound (LB) within the B&B algorithm. Since lower bounds directly determine pruning efficiency, their quality is expected to critically influence the solution times.

The following combinations are tested (single representative instance per configuration):

- Number of operations: 15
- Number of workstations: 4 and 6
- Number of operations per block: 2 and 4
- Number of blocks per workstation: 2 and 4
- Maintenance type: Case 1 and Case 2
- Maintenance probabilities: Identical probabilities
- Precedence structure: Tight and loose
- Operation time distributions: U[10, 20]

The results are provided in Table 43.

Table 43: The results of B&B algorithm (With/Without LB)

| <i>m</i> | <i>n</i> ₀ | <i>m</i> ₀ | Case Type | With <i>LB</i> | | | | Without <i>LB</i> | | | |
|----------|-----------------------|-----------------------|-----------|----------------|----------|----------------|-------------|-------------------|----------|----------------|-------------|
| | | | | <i>UB</i> | CPU Time | Best Sol. Time | Total Node | <i>UB</i> | CPU Time | Best Sol. Time | Total Node |
| 4 | 2 | 2 | 1 | 42.15 | 129.60 | 21.12 | 5,118,076 | 42.15 | 1234.75 | 147.74 | 50,098,372 |
| 4 | 2 | 2 | 2 | 219.92 | 1364.82 | 161.53 | 50,097,152 | 219.92 | 1254.99 | 146.44 | 50,097,152 |
| 4 | 2 | 4 | 1 | 43.46 | 7200.00 | 929.03 | 201,161,409 | 45.20 | 7200.00 | 3867.72 | 226,324,623 |
| 4 | 2 | 4 | 2 | 214.58 | 7200.00 | 1850.72 | 207,107,769 | 214.58 | 7200.00 | 2050.61 | 226,602,441 |
| 4 | 4 | 2 | 1 | 34.13 | 279.90 | 93.56 | 11,724,203 | 34.13 | 7200.00 | 4407.54 | 344,241,503 |
| 4 | 4 | 2 | 2 | 195.27 | 7154.45 | 303.16 | 293,206,220 | 195.27 | 7200.00 | 279.07 | 333,328,950 |
| 4 | 4 | 4 | 1 | 35.50 | 7200.00 | 1927.79 | 229,088,430 | 39.50 | 7200.00 | 570.97 | 247,555,924 |
| 4 | 4 | 4 | 2 | 205.70 | 7200.00 | 138.26 | 228,987,382 | 205.70 | 7200.00 | 144.56 | 247,275,682 |
| 6 | 2 | 2 | 1 | 34.14 | 647.38 | 133.32 | 22,955,824 | 34.97 | 7200.00 | 3023.21 | 313,011,920 |
| 6 | 2 | 2 | 2 | 220.64 | 7200.00 | 3759.31 | 116,051,755 | 220.64 | 7200.00 | 3621.93 | 234,538,587 |
| 6 | 2 | 4 | 1 | 34.97 | 7200.00 | 6864.43 | 201,811,680 | 38.68 | 7200.00 | 1281.78 | 240,358,862 |
| 6 | 2 | 4 | 2 | 214.58 | 7200.00 | 5561.55 | 107,855,424 | 214.58 | 7200.00 | 5966.46 | 190,803,913 |
| 6 | 4 | 2 | 1 | 28.01 | 57.74 | 8.58 | 2,142,825 | 28.01 | 7200.00 | 226.34 | 338,515,837 |
| 6 | 4 | 2 | 2 | 195.27 | 418.15 | 418.15 | 7,465,709 | 212.89 | 7200.00 | 411.84 | 263,902,311 |
| 6 | 4 | 4 | 1 | 28.01 | 5290.18 | 916.76 | 149,286,198 | 32.28 | 7200.00 | 1865.59 | 259,711,598 |
| 6 | 4 | 4 | 2 | 205.70 | 259.043 | 259.043 | 3,859,790 | 211.69 | 7200.00 | 203.90 | 194,678,421 |

The inclusion of the LB drastically reduces the size of the search tree. In every instance, the "With LB" configuration explores fewer nodes than the "Without LB" configuration as expected. This order of magnitude reduction confirms that the LB is highly effective at pruning unpromising branches early in the search process, preventing the algorithm from wasting computational resources on infeasible or suboptimal sub-problems.

The reduction in the search space translates directly into improved computational times. Several instances that reach the termination limit without the LB are solved to optimality well within the limit when the LB is applied. A notable example is the instance with $m = 4$, $n_0 = 4$, $m_0 = 2$, case 1; "Without LB", the algorithm times out at 7200 seconds, but "With LB", it solves the problem in just 279.90 seconds. Consequently, LB serves not just as an optimization strategy, but a fundamental requirement for making the B&B algorithm tractable.

For harder instances where both configurations hit the time limit, the "With LB" approach often yields a better UB. For instance, in the setup $m = 4$, $n_0 = 2$, $m_0 = 4$, case 1; the "With LB" achieves a UB value of 43.46 compared to 45.20 for the "Without LB". This suggests that the LB mechanism helps the algorithm focus its search on high-quality regions of the solution space faster, finding better feasible solutions even when optimality cannot be proven.

6.3.3 Model- and Rule-based Heuristics

Preliminary experiments for the matheuristics aim to understand the following aspects:

1. Behavior of the block formation models
2. Effectiveness of improvement procedures

We use the following setup for the preliminary experiments for Model- and Rule-based heuristics:

- Number of operations: 15, 20, 25, 30, 40, 50
- Number of workstations: 4 and 6
- Number of operations per block: 4
- Number of blocks per workstation: 4
- Maintenance type: Case 1
- Maintenance probabilities: Identical probabilities

- Precedence structure: Tight and loose
- Operation time distributions: $U[10, 20]$
- Earliest and latest workstation parameters (E_i and L_i): Included

First of all, we analyze the results of the mathematical model-based heuristics without any improvement procedures and report the results of CPU times, deviations from optimal and frequency bests, in Table 44, Table 45 and Table 46, respectively.

Table 44: CPU times of Model-based heuristics

| n | m | Setup | B0-W | | B1-W | | B2-W | | B3-W | |
|-----|-----|-------|-------|-------|------|-------|-------|-------|--------|--------|
| | | | Avg | Max | Avg | Max | Avg | Max | Avg | Max |
| 15 | 4 | T | 0.04 | 0.09 | 0.03 | 0.05 | 0.12 | 0.30 | 1.39 | 5.78 |
| 15 | 4 | L | 0.06 | 0.19 | 0.04 | 0.13 | 0.02 | 0.03 | 4.56 | 7.56 |
| 15 | 6 | T | 0.12 | 0.25 | 0.19 | 0.81 | 0.54 | 1.94 | 12.78 | 66.61 |
| 15 | 6 | L | 0.88 | 2.20 | 1.45 | 5.74 | 0.09 | 0.11 | 41.58 | 97.80 |
| 20 | 4 | T | 0.03 | 0.05 | 0.03 | 0.05 | 0.16 | 0.31 | 7.19 | 29.27 |
| 20 | 4 | L | 0.08 | 0.20 | 0.05 | 0.12 | 0.03 | 0.05 | 176.25 | 900.02 |
| 20 | 6 | T | 0.13 | 0.59 | 0.14 | 0.66 | 1.57 | 5.28 | 102.76 | 527.59 |
| 20 | 6 | L | 14.39 | 73.36 | 4.86 | 14.14 | 0.10 | 0.14 | 603.13 | 900.08 |
| 25 | 4 | T | 0.07 | 0.16 | 0.07 | 0.13 | 0.14 | 0.36 | 6.90 | 36.53 |
| 25 | 4 | L | 0.13 | 0.17 | 0.12 | 0.16 | 0.12 | 0.77 | 757.80 | 900.05 |
| 25 | 6 | T | 0.17 | 0.39 | 0.17 | 0.31 | 2.23 | 6.86 | 235.90 | 900.05 |
| 25 | 6 | L | 2.45 | 5.53 | 8.05 | 58.67 | 0.12 | 0.19 | 900.03 | 900.05 |
| 30 | 4 | T | 0.07 | 0.20 | 0.08 | 0.23 | 0.18 | 0.41 | 60.68 | 538.66 |
| 30 | 4 | L | 0.13 | 0.17 | 0.12 | 0.19 | 0.08 | 0.23 | 900.00 | 900.02 |
| 30 | 6 | T | 0.27 | 0.45 | 0.21 | 0.36 | 4.86 | 13.80 | 447.50 | 900.02 |
| 30 | 6 | L | 5.03 | 44.84 | 4.18 | 23.06 | 0.14 | 0.20 | 900.04 | 900.16 |
| 40 | 4 | T | 0.19 | 0.91 | 0.12 | 0.38 | 0.14 | 0.30 | 498.51 | 900.02 |
| 40 | 4 | L | 0.08 | 0.16 | 0.06 | 0.11 | 0.08 | 0.13 | 900.00 | 900.02 |
| 40 | 6 | T | 0.34 | 0.52 | 0.33 | 0.62 | 10.19 | 98.63 | 900.02 | 900.03 |
| 40 | 6 | L | 0.48 | 1.94 | 0.48 | 1.36 | 0.25 | 0.38 | 900.04 | 900.08 |
| 50 | 4 | T | 0.12 | 0.17 | 0.13 | 0.19 | 0.11 | 0.16 | 900.00 | 900.02 |
| 50 | 4 | L | 0.13 | 0.27 | 0.10 | 0.20 | 0.09 | 0.11 | 900.01 | 900.02 |
| 50 | 6 | T | 0.41 | 0.61 | 0.38 | 0.61 | 0.42 | 0.94 | 900.04 | 900.06 |
| 50 | 6 | L | 0.37 | 0.58 | 0.37 | 0.67 | 0.39 | 0.53 | 900.04 | 900.11 |

The results in Table 44 indicate that, other than the Model B3-based heuristic, heuristics demonstrate high computational efficiency, with average CPU times remaining below 1 second for the vast majority of instances, even as the number of operations scales to 50. This suggests these algorithms are robustly scalable. For the Model B3-based heuristic, the termination limit of 900 seconds is reached for

loose instances starting at $n = 25$ and for all instances when the number of operations reaches 50.

Table 45: Percent deviation from optimal of Model-based heuristics

| n | m | Setup | B0-W | | B1-W | | B2-W | | B3-W | |
|-----|-----|-------|------|-----|------|-----|------|-----|------|-----|
| | | | Avg | Max | Avg | Max | Avg | Max | Avg | Max |
| 15 | 4 | T | 61 | 96 | 59 | 100 | 23 | 64 | 26 | 70 |
| 15 | 4 | L | 90 | 112 | 81 | 108 | 12 | 56 | 21 | 55 |
| 15 | 6 | T | 48 | 97 | 50 | 75 | 31 | 101 | 34 | 96 |
| 15 | 6 | L | 51 | 63 | 44 | 56 | 3 | 4 | 21 | 55 |
| 20 | 4 | T | 50 | 94 | 36 | 63 | 29 | 58 | 26 | 58 |
| 20 | 4 | L | 71 | 84 | 74 | 88 | 23 | 34 | 15 | 58 |
| 20 | 6 | T | 68 | 151 | 58 | 116 | 42 | 86 | 42 | 86 |
| 20 | 6 | L | 70 | 96 | 72 | 91 | 21 | 54 | 27 | 82 |
| 25 | 4 | T | 40 | 77 | 35 | 74 | 22 | 65 | 18 | 73 |
| 25 | 4 | L | 59 | 77 | 67 | 87 | 17 | 47 | 21 | 41 |
| 25 | 6 | T | 57 | 94 | 48 | 80 | 55 | 117 | 46 | 96 |
| 25 | 6 | L | 73 | 84 | 78 | 98 | 30 | 40 | 43 | 82 |
| 30 | 4 | T | 51 | 78 | 50 | 79 | 24 | 76 | 20 | 52 |
| 30 | 4 | L | 69 | 86 | 68 | 81 | 39 | 92 | 30 | 69 |
| 30 | 6 | T | 65 | 107 | 54 | 89 | 33 | 75 | 32 | 80 |
| 30 | 6 | L | 74 | 121 | 76 | 113 | 71 | 122 | 39 | 96 |
| 40 | 4 | T | 39 | 47 | 38 | 44 | 24 | 47 | 15 | 38 |
| 40 | 4 | L | 47 | 55 | 51 | 59 | 24 | 48 | 21 | 43 |
| 40 | 6 | T | 64 | 89 | 59 | 87 | 31 | 45 | 31 | 76 |
| 40 | 6 | L | 72 | 102 | 70 | 94 | 56 | 89 | 35 | 68 |
| 50 | 4 | T | 28 | 33 | 27 | 33 | 21 | 34 | 9 | 17 |
| 50 | 4 | L | 29 | 35 | 30 | 37 | 25 | 31 | 13 | 19 |
| 50 | 6 | T | 56 | 72 | 57 | 77 | 42 | 81 | 33 | 55 |
| 50 | 6 | L | 63 | 74 | 66 | 77 | 57 | 72 | 38 | 55 |

The results in Table 45 indicate that the computation times of Model B0- and B1-based heuristics comes at the cost of solution quality, with average deviations often exceeding 50% and reaching as high as 90% for certain loose setups. The Model B3-based heuristic generally offers the lowest deviation from optimality, identifying it as the most robust heuristic in terms of solution quality, particularly for larger instances. The Model B2-based heuristic emerges as a notable compromise, offering significantly lower deviations than Model B0- and B1-based heuristics while retaining near-instantaneous solution times. Notably, for small-sized instances where the number of operations is 15, the Model B2-based heuristic performs better than the Model B3-based heuristic.

Table 46 presents the comparative efficacy of the four model-based heuristics by analyzing two key frequency metrics: Frequency Best, which counts how often a specific heuristic achieves the best solution among the four variants, and Frequency Optimal, which counts how often the heuristics match the best solution found by the exact SP model. It is important to note that for this analysis, the SP model's solution is used as the benchmark for optimality.

Table 46: Frequency best and optimal solutions of Model-based heuristics

| n | m | Setup | B-best | B0-W | | B1-W | | B2-W | | B3-W | |
|-----|-----|-------|--------|------|------|------|------|------|------|------|------|
| | | | OPT | OPT | BEST | OPT | BEST | OPT | BEST | OPT | BEST |
| 15 | 4 | T | 3 | 0 | 0 | 0 | 1 | 1 | 5 | 2 | 3 |
| 15 | 4 | L | 4 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 4 |
| 15 | 6 | T | 2 | 0 | 2 | 0 | 1 | 0 | 3 | 2 | 3 |
| 15 | 6 | L | 4 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 4 |
| 20 | 4 | T | 1 | 0 | 1 | 0 | 2 | 1 | 2 | 1 | 5 |
| 20 | 4 | L | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 7 |
| 20 | 6 | T | 2 | 0 | 1 | 0 | 3 | 1 | 4 | 2 | 4 |
| 20 | 6 | L | 3 | 0 | 0 | 0 | 0 | 0 | 4 | 3 | 3 |
| 25 | 4 | T | 2 | 0 | 1 | 0 | 0 | 1 | 4 | 2 | 7 |
| 25 | 4 | L | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 3 |
| 25 | 6 | T | 1 | 0 | 0 | 0 | 2 | 1 | 3 | 1 | 5 |
| 25 | 6 | L | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 3 |
| 30 | 4 | T | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 7 |
| 30 | 4 | L | 1 | 0 | 0 | 0 | 0 | 0 | 3 | 1 | 5 |
| 30 | 6 | T | 1 | 0 | 1 | 0 | 0 | 0 | 4 | 1 | 5 |
| 30 | 6 | L | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 4 |
| 40 | 4 | T | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 8 |
| 40 | 4 | L | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 6 |
| 40 | 6 | T | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 6 |
| 40 | 6 | L | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 |
| 50 | 4 | T | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 10 |
| 50 | 4 | L | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 |
| 50 | 6 | T | 1 | 0 | 0 | 0 | 0 | 0 | 3 | 1 | 7 |
| 50 | 6 | L | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 9 |

The Model B3-based heuristic exhibits a clear and growing dominance in solution quality as problem complexity increases. This trend is particularly evident in larger problem sizes where Model B3-based heuristic consistently identifies the best solution (or ties for the best) more frequently than any other variant. For example, in the largest configuration ($n = 50, m = 4, \text{loose}$), the Model B3-based heuristic achieves the best solution in 10 out of 10 instances, whereas the Model B2-based heuristic fails to find the best solution in any instance (0 out of 10). Furthermore,

for smaller problem sizes, the Model B3-based heuristic frequently reaches the optimal solution, validating that its underlying logic is significantly more robust at navigating the solution space of complex, large-sized problems compared to its counterparts.

While the Model B3-based heuristic dominates overall, the Model B2-based heuristic demonstrates remarkable competitiveness in small and medium-sized instances. In specific configurations, such as $(n = 15, m = 4, \text{tight})$, the Model B2-based heuristic actually outperforms the Model B3-based heuristic in terms of "Frequency Best". This indicates that for moderately sized problems, the additional computational overhead of the Model B3-based heuristic may not always be necessary to find high-quality solutions. Consequently, the Model B2-based heuristic serves as a viable middle-ground approach, offering a balance between solution quality and computational time.

The results definitively show that the Model B0- and B1-based heuristics are ineffective at finding high-quality solutions. Their columns are predominantly filled with zeros, indicating they rarely, if ever, outperform the B2- or B3-based heuristics. This corroborates the findings from previous sections regarding CPU times and deviations, confirming that while Model B0- and B1-based heuristics are computationally inexpensive, they lack the algorithmic sophistication required to compete with Model B2- and B3-based heuristic in terms of solution quality.

Then, we investigated that the results of the mathematical model-based heuristics after applying the improvement procedures separately and report the results of CPU times, deviations from optimal, in Table 47, 48 and Table 49,50, respectively.

Table 47: CPU times of improvement procedure 1 on Model-based heuristics

| n | m | Setup | B0-W-IP1 | | B1-W-IP1 | | B2-W-IP1 | | B3-W-IP1 | |
|-----|-----|-------|----------|------|----------|------|----------|------|----------|-------|
| | | | Avg | Max | Avg | Max | Avg | Max | Avg | Max |
| 15 | 4 | T | 0.44 | 0.65 | 0.44 | 0.85 | 0.32 | 0.45 | 1.55 | 5.89 |
| 15 | 4 | L | 0.48 | 0.69 | 0.47 | 0.88 | 0.24 | 0.51 | 4.74 | 7.68 |
| 15 | 6 | T | 0.53 | 0.77 | 0.58 | 1.06 | 1.32 | 3.08 | 13.28 | 67.30 |
| 15 | 6 | L | 1.73 | 2.28 | 1.66 | 2.46 | 0.77 | 1.31 | 42.08 | 98.17 |

| <i>n</i> | <i>m</i> | Setup | B0-W-IP1 | | B1-W-IP1 | | B2-W-IP1 | | B3-W-IP1 | |
|----------|----------|-------|----------|-------|----------|-------|----------|-------|----------|--------|
| | | | Avg | Max | Avg | Max | Avg | Max | Avg | Max |
| 20 | 4 | T | 2.65 | 4.16 | 3.05 | 8.00 | 0.38 | 0.58 | 7.36 | 29.46 |
| 20 | 4 | L | 0.54 | 0.94 | 0.50 | 1.20 | 0.25 | 0.43 | 176.38 | 900.14 |
| 20 | 6 | T | 0.70 | 1.21 | 0.70 | 1.12 | 2.18 | 5.99 | 103.45 | 530.22 |
| 20 | 6 | L | 1.62 | 3.06 | 1.17 | 3.18 | 1.10 | 1.70 | 603.80 | 900.84 |
| 25 | 4 | T | 17.70 | 76.84 | 7.61 | 15.46 | 0.40 | 0.63 | 7.05 | 36.66 |
| 25 | 4 | L | 0.63 | 0.91 | 0.65 | 1.20 | 0.58 | 1.40 | 758.14 | 901.14 |
| 25 | 6 | T | 1.15 | 1.67 | 1.02 | 1.62 | 2.95 | 7.59 | 236.41 | 900.44 |
| 25 | 6 | L | 1.89 | 4.38 | 2.13 | 4.71 | 1.27 | 2.53 | 900.99 | 902.00 |
| 30 | 4 | T | 5.67 | 8.10 | 11.05 | 59.46 | 0.55 | 1.05 | 60.87 | 538.80 |
| 30 | 4 | L | 0.87 | 1.40 | 0.96 | 1.20 | 0.62 | 0.87 | 900.29 | 901.12 |
| 30 | 6 | T | 1.12 | 1.94 | 1.63 | 2.57 | 6.02 | 15.15 | 448.34 | 901.57 |
| 30 | 6 | L | 3.50 | 5.92 | 3.38 | 6.77 | 2.00 | 3.90 | 901.13 | 901.91 |
| 40 | 4 | T | 8.05 | 46.77 | 7.91 | 26.59 | 0.56 | 1.07 | 498.74 | 900.36 |
| 40 | 4 | L | 1.31 | 2.57 | 1.26 | 1.83 | 0.66 | 1.46 | 900.32 | 900.85 |
| 40 | 6 | T | 1.31 | 3.17 | 1.03 | 1.93 | 11.50 | 99.89 | 900.90 | 901.51 |
| 40 | 6 | L | 3.38 | 6.52 | 3.96 | 8.82 | 2.20 | 2.89 | 900.93 | 901.93 |
| 50 | 4 | T | 3.29 | 5.26 | 4.45 | 10.03 | 0.90 | 2.23 | 900.21 | 900.39 |
| 50 | 4 | L | 2.05 | 2.96 | 1.55 | 2.19 | 1.13 | 1.71 | 900.24 | 900.35 |
| 50 | 6 | T | 1.31 | 2.61 | 1.63 | 2.79 | 3.26 | 6.69 | 901.34 | 902.66 |
| 50 | 6 | L | 6.17 | 11.93 | 5.55 | 8.87 | 3.41 | 8.18 | 901.49 | 904.48 |

Table 48: CPU times of improvement procedure 2 on Model-based heuristics

| <i>n</i> | <i>m</i> | Setup | B0-W-IP2 | | B1-W-IP2 | | B2-W-IP2 | | B3-W-IP2 | |
|----------|----------|-------|----------|--------|----------|--------|----------|--------|----------|---------|
| | | | Avg | Max | Avg | Max | Avg | Max | Avg | Max |
| 15 | 4 | T | 0.17 | 0.31 | 0.17 | 0.39 | 0.36 | 0.53 | 1.63 | 5.92 |
| 15 | 4 | L | 0.15 | 0.27 | 0.15 | 0.22 | 0.21 | 0.27 | 4.75 | 7.75 |
| 15 | 6 | T | 0.33 | 0.52 | 0.37 | 1.01 | 0.92 | 2.22 | 13.16 | 66.84 |
| 15 | 6 | L | 1.02 | 2.33 | 1.60 | 5.93 | 0.79 | 1.36 | 42.28 | 98.14 |
| 20 | 4 | T | 0.18 | 0.27 | 0.21 | 0.43 | 0.49 | 0.70 | 7.53 | 29.71 |
| 20 | 4 | L | 0.21 | 0.35 | 0.15 | 0.21 | 0.55 | 0.79 | 176.76 | 900.54 |
| 20 | 6 | T | 0.43 | 1.28 | 0.44 | 1.23 | 2.49 | 5.53 | 103.68 | 528.19 |
| 20 | 6 | L | 14.70 | 73.71 | 5.18 | 14.42 | 1.19 | 5.58 | 604.22 | 905.56 |
| 25 | 4 | T | 0.48 | 0.70 | 0.44 | 0.66 | 0.63 | 1.52 | 7.40 | 37.51 |
| 25 | 4 | L | 0.57 | 0.80 | 0.55 | 0.68 | 1.27 | 3.86 | 758.95 | 903.82 |
| 25 | 6 | T | 0.78 | 1.42 | 0.75 | 1.28 | 3.25 | 7.35 | 236.92 | 900.41 |
| 25 | 6 | L | 3.26 | 6.22 | 8.83 | 59.43 | 0.87 | 2.87 | 900.78 | 902.80 |
| 30 | 4 | T | 0.36 | 0.79 | 0.36 | 0.76 | 0.60 | 1.71 | 61.11 | 540.22 |
| 30 | 4 | L | 0.71 | 1.09 | 0.66 | 1.02 | 4.22 | 30.51 | 904.14 | 930.45 |
| 30 | 6 | T | 1.07 | 1.85 | 0.96 | 1.73 | 5.76 | 14.56 | 448.41 | 901.76 |
| 30 | 6 | L | 5.85 | 45.63 | 5.04 | 23.80 | 91.39 | 900.14 | 991.29 | 1800.02 |
| 40 | 4 | T | 0.66 | 1.42 | 0.71 | 1.24 | 1.09 | 3.34 | 499.46 | 903.19 |
| 40 | 4 | L | 0.41 | 0.61 | 0.42 | 0.62 | 18.60 | 77.15 | 918.52 | 977.11 |
| 40 | 6 | T | 1.11 | 1.32 | 1.17 | 2.40 | 11.57 | 99.48 | 901.40 | 904.82 |
| 40 | 6 | L | 1.09 | 2.45 | 1.12 | 1.78 | 3.62 | 18.76 | 903.41 | 918.54 |
| 50 | 4 | T | 18.09 | 171.62 | 21.84 | 171.04 | 48.44 | 198.14 | 948.33 | 1098.08 |
| 50 | 4 | L | 91.37 | 900.03 | 38.30 | 353.19 | 333.63 | 900.09 | 1233.54 | 1800.02 |
| 50 | 6 | T | 1.14 | 1.32 | 1.06 | 1.50 | 11.56 | 96.48 | 911.18 | 996.18 |
| 50 | 6 | L | 1.08 | 1.21 | 1.11 | 1.35 | 102.90 | 903.53 | 1002.55 | 1803.12 |

The results in Table 47 indicate that the computational time for Improvement Procedure 1 (IP1) is negligible for all mathematical model-based heuristics, average CPU times remain under a few seconds even for large-sized instances. This efficiency is expected due to the polynomial structure of the procedure, which allows it to scale gracefully as problem size increases.

Results in Table 48 presents that the computational time required for Improvement Procedure 2 (IP2) increases as the number of operations increases. While efficient for smaller problems, the procedure becomes computationally intensive for larger datasets, specifically after $n = 30$. For some setups, the Model B3-based heuristic hits the cumulative termination limit.

Table 49: Percent deviations of improvement procedure 1 on Model-based heuristics

| n | m | Setup | B0-W-IP1 | | B1-W-IP1 | | B2-W-IP1 | | B3-W-IP1 | |
|-----|-----|-------|----------|-----|----------|-----|----------|-----|----------|-----|
| | | | Avg | Max | Avg | Max | Avg | Max | Avg | Max |
| 15 | 4 | T | 15 | 32 | 24 | 77 | 14 | 53 | 21 | 70 |
| 15 | 4 | L | 18 | 46 | 22 | 59 | 3 | 23 | 13 | 47 |
| 15 | 6 | T | 11 | 50 | 10 | 24 | 16 | 54 | 26 | 91 |
| 15 | 6 | L | 5 | 23 | 3 | 24 | 1 | 2 | 13 | 47 |
| 20 | 4 | T | 17 | 56 | 15 | 32 | 22 | 58 | 23 | 58 |
| 20 | 4 | L | 20 | 28 | 24 | 69 | 10 | 24 | 12 | 50 |
| 20 | 6 | T | 32 | 85 | 37 | 109 | 35 | 86 | 35 | 86 |
| 20 | 6 | L | 5 | 21 | 12 | 43 | 10 | 40 | 16 | 45 |
| 25 | 4 | T | 15 | 36 | 13 | 30 | 12 | 50 | 17 | 69 |
| 25 | 4 | L | 13 | 35 | 20 | 59 | 7 | 11 | 14 | 41 |
| 25 | 6 | T | 37 | 87 | 24 | 50 | 48 | 105 | 45 | 92 |
| 25 | 6 | L | 23 | 39 | 25 | 55 | 15 | 35 | 29 | 70 |
| 30 | 4 | T | 25 | 65 | 18 | 53 | 17 | 52 | 18 | 52 |
| 30 | 4 | L | 24 | 63 | 15 | 57 | 26 | 71 | 24 | 66 |
| 30 | 6 | T | 19 | 29 | 22 | 57 | 26 | 61 | 29 | 76 |
| 30 | 6 | L | 26 | 73 | 26 | 58 | 39 | 99 | 27 | 52 |
| 40 | 4 | T | 17 | 27 | 18 | 29 | 16 | 40 | 13 | 31 |
| 40 | 4 | L | 19 | 35 | 28 | 43 | 16 | 33 | 18 | 40 |
| 40 | 6 | T | 32 | 47 | 25 | 66 | 26 | 45 | 29 | 74 |
| 40 | 6 | L | 40 | 80 | 30 | 85 | 34 | 56 | 31 | 63 |
| 50 | 4 | T | 10 | 21 | 12 | 21 | 12 | 23 | 9 | 13 |
| 50 | 4 | L | 17 | 22 | 17 | 23 | 14 | 22 | 11 | 19 |
| 50 | 6 | T | 28 | 56 | 32 | 62 | 23 | 57 | 29 | 55 |
| 50 | 6 | L | 25 | 48 | 37 | 60 | 37 | 57 | 37 | 52 |

Table 50: Percent deviations of improvement procedure 2 on Model-based heuristics

| n | m | Setup | B0-W-IP2 | | B1-W-IP2 | | B2-W-IP2 | | B3-W-IP2 | |
|-----|-----|-------|----------|-----|----------|-----|----------|-----|----------|-----|
| | | | Avg | Max | Avg | Max | Avg | Max | Avg | Max |
| 15 | 4 | T | 44 | 78 | 40 | 84 | 4 | 22 | 1 | 6 |
| 15 | 4 | L | 72 | 104 | 58 | 101 | 1 | 2 | 0 | 1 |
| 15 | 6 | T | 30 | 52 | 32 | 49 | 9 | 40 | 0 | 1 |
| 15 | 6 | L | 40 | 56 | 36 | 49 | 1 | 2 | 0 | 1 |
| 20 | 4 | T | 22 | 71 | 15 | 30 | 4 | 9 | 2 | 9 |
| 20 | 4 | L | 56 | 77 | 63 | 80 | 6 | 12 | 5 | 18 |
| 20 | 6 | T | 43 | 106 | 39 | 77 | 3 | 15 | 6 | 25 |
| 20 | 6 | L | 62 | 92 | 63 | 82 | 1 | 3 | 3 | 31 |
| 25 | 4 | T | 19 | 47 | 16 | 37 | 2 | 4 | 1 | 6 |
| 25 | 4 | L | 43 | 61 | 52 | 64 | 7 | 12 | 2 | 7 |
| 25 | 6 | T | 25 | 53 | 26 | 42 | 9 | 30 | 10 | 33 |
| 25 | 6 | L | 61 | 81 | 70 | 92 | 8 | 28 | 11 | 34 |
| 30 | 4 | T | 29 | 69 | 24 | 57 | 4 | 9 | 5 | 16 |
| 30 | 4 | L | 42 | 63 | 46 | 64 | 7 | 19 | 4 | 14 |
| 30 | 6 | T | 38 | 71 | 31 | 56 | 7 | 16 | 6 | 24 |
| 30 | 6 | L | 57 | 86 | 65 | 92 | 18 | 29 | 10 | 33 |
| 40 | 4 | T | 23 | 38 | 18 | 29 | 6 | 13 | 3 | 6 |
| 40 | 4 | L | 35 | 43 | 33 | 48 | 11 | 19 | 4 | 23 |
| 40 | 6 | T | 44 | 70 | 36 | 65 | 11 | 35 | 12 | 34 |
| 40 | 6 | L | 52 | 72 | 49 | 65 | 24 | 46 | 5 | 23 |
| 50 | 4 | T | 18 | 24 | 16 | 26 | 8 | 27 | 3 | 9 |
| 50 | 4 | L | 18 | 25 | 16 | 24 | 11 | 26 | 3 | 9 |
| 50 | 6 | T | 39 | 57 | 42 | 67 | 18 | 30 | 14 | 31 |
| 50 | 6 | L | 47 | 65 | 45 | 69 | 26 | 39 | 11 | 34 |

The results in Table 49 and Table 50 indicate that Model B0- and B1-based heuristics still exhibit the highest deviations, even after improvement procedures are applied. Average deviations for these models frequently exceed 50%, and in certain loose setups maximum deviations reach as high as 109% and 106%. This suggests that while these models are useful for rapid solution generation, they remain less reliable for high-precision requirements.

In contrast, Model B2- and Model B3-based heuristics consistently offer the lowest deviations from optimality across the vast majority of problem instances. Specifically, under Improvement Procedure 2, the Model B3-based heuristic frequently achieves single-digit or near-zero deviations identifying it as the most robust heuristic in terms of solution quality.

After observing the promising results of the Model B2- and B3-based heuristics, a new approach is proposed in which Improvement Procedure 1 (IP1) is applied first, followed by Improvement Procedure 2 (IP2). The resulting models are named as follows: when the Model B2-based heuristic is used and both procedures are applied sequentially, the approach is referred to as the Model B2–IP12-based heuristic. Similarly, when this sequence is applied to the Model B3-based heuristic, it is referred to as the Model B3–IP12-based heuristic.

We analyze the results of the Model B2- and B3-based variants heuristics and report the results of CPU times, deviations from optimal and frequency best, in Table 51, Table 52 and Table 53, respectively.

Table 51: CPU times of Model-based heuristics after improvement procedures

| n | m | Setup | B2-W | | | | | | B3-W | | | | | |
|----|---|-------|-------|-------|--------|--------|--------|--------|--------|--------|---------|---------|---------|---------|
| | | | IP1 | | IP2 | | IP12 | | IP1 | | IP2 | | IP12 | |
| | | | Avg | Max | Avg | Max | Avg | Max | Avg | Max | Avg | Max | Avg | Max |
| 15 | 4 | T | 0.32 | 0.45 | 0.36 | 0.53 | 0.55 | 0.68 | 1.55 | 5.89 | 1.63 | 5.92 | 1.87 | 6.62 |
| 15 | 4 | L | 0.24 | 0.51 | 0.21 | 0.27 | 0.43 | 0.67 | 4.74 | 7.68 | 4.75 | 7.75 | 4.99 | 7.84 |
| 15 | 6 | T | 1.32 | 3.08 | 0.92 | 2.22 | 1.71 | 3.32 | 13.28 | 67.30 | 13.16 | 66.84 | 18.59 | 111.56 |
| 15 | 6 | L | 0.77 | 1.31 | 0.79 | 1.36 | 1.47 | 2.10 | 42.08 | 98.17 | 42.28 | 98.14 | 44.40 | 98.49 |
| 20 | 4 | T | 0.38 | 0.58 | 0.49 | 0.70 | 0.71 | 0.95 | 7.36 | 29.46 | 7.53 | 29.71 | 8.75 | 30.26 |
| 20 | 4 | L | 0.25 | 0.43 | 0.55 | 0.79 | 0.77 | 1.01 | 176.38 | 900.14 | 176.76 | 900.54 | 207.72 | 900.62 |
| 20 | 6 | T | 2.18 | 5.99 | 2.49 | 5.53 | 3.11 | 6.24 | 103.45 | 530.22 | 103.68 | 528.19 | 104.85 | 534.00 |
| 20 | 6 | L | 1.10 | 1.70 | 1.19 | 5.58 | 2.19 | 6.73 | 603.80 | 900.84 | 604.22 | 905.56 | 604.64 | 903.22 |
| 25 | 4 | T | 0.40 | 0.63 | 0.63 | 1.52 | 0.89 | 2.07 | 7.05 | 36.66 | 7.40 | 37.51 | 7.54 | 38.91 |
| 25 | 4 | L | 0.58 | 1.40 | 1.27 | 3.86 | 1.72 | 4.52 | 758.14 | 901.14 | 758.95 | 903.82 | 764.94 | 945.13 |
| 25 | 6 | T | 2.95 | 7.59 | 3.25 | 7.35 | 3.97 | 8.08 | 236.41 | 900.44 | 236.92 | 900.41 | 237.26 | 900.86 |
| 25 | 6 | L | 1.27 | 2.53 | 0.87 | 2.87 | 2.02 | 3.69 | 900.99 | 902.00 | 900.78 | 902.80 | 904.11 | 919.08 |
| 30 | 4 | T | 0.55 | 1.05 | 0.60 | 1.71 | 0.98 | 2.55 | 60.87 | 538.80 | 61.11 | 540.22 | 61.67 | 543.67 |
| 30 | 4 | L | 0.62 | 0.87 | 4.22 | 30.51 | 4.76 | 30.86 | 900.29 | 901.12 | 904.14 | 930.45 | 902.75 | 914.44 |
| 30 | 6 | T | 6.02 | 15.15 | 5.76 | 14.56 | 6.92 | 15.92 | 448.34 | 901.57 | 448.41 | 901.76 | 449.23 | 902.40 |
| 30 | 6 | L | 2.00 | 3.90 | 91.39 | 900.14 | 93.25 | 901.90 | 901.13 | 901.91 | 991.29 | 1800.02 | 904.27 | 922.66 |
| 40 | 4 | T | 0.56 | 1.07 | 1.09 | 3.34 | 1.51 | 4.27 | 498.74 | 900.36 | 499.46 | 903.19 | 500.28 | 908.96 |
| 40 | 4 | L | 0.66 | 1.46 | 18.60 | 77.15 | 19.17 | 77.79 | 900.32 | 900.85 | 918.52 | 977.11 | 1061.50 | 1800.41 |
| 40 | 6 | T | 11.50 | 99.89 | 11.57 | 99.48 | 12.87 | 100.74 | 900.90 | 901.51 | 901.40 | 904.82 | 901.84 | 902.83 |
| 40 | 6 | L | 2.20 | 2.89 | 3.62 | 18.76 | 5.57 | 20.39 | 900.93 | 901.93 | 903.41 | 918.54 | 931.62 | 1047.25 |
| 50 | 4 | T | 0.90 | 2.23 | 48.44 | 198.14 | 49.23 | 198.39 | 900.21 | 900.39 | 948.33 | 1098.08 | 1061.00 | 1697.98 |
| 50 | 4 | L | 1.13 | 1.71 | 333.63 | 900.09 | 334.67 | 901.66 | 900.24 | 900.35 | 1233.54 | 1800.02 | 1233.62 | 1800.23 |
| 50 | 6 | T | 3.26 | 6.69 | 11.56 | 96.48 | 14.40 | 102.84 | 901.34 | 902.66 | 911.18 | 996.18 | 918.97 | 979.81 |
| 50 | 6 | L | 3.41 | 8.18 | 102.90 | 903.53 | 105.93 | 908.05 | 901.49 | 904.48 | 1002.55 | 1803.12 | 906.25 | 916.32 |

The results in Table 51 indicate that the application of improvement procedures introduces a distinct computational hierarchy. Improvement Procedure 1 (IP1) remains computationally inexpensive for the Model B2-based heuristic, adding negligible time to the base models. However, Improvement Procedure 2 (IP2) significantly increases CPU time, particularly for the Model B3 variant. For large-

sized, loose instances, the Model B3–IP12-based heuristic reaches the cumulative time limit of 1800 seconds. Conversely, the Model B2–IP12 heuristic offers a balanced profile; while it is slower than the Model B2-based heuristic, it remains significantly faster than Model B3-based variants making it a viable alternative for time-constrained scenarios.

Table 52: Percent deviations of Model B2- and B3-based heuristics after improvement procedures

| n | m | Setup | B2-W | | | | | | B3-W | | | | | |
|----|---|-------|------|-----|-----|-----|------|-----|------|-----|-----|-----|------|-----|
| | | | IP1 | | IP2 | | IP12 | | IP1 | | IP2 | | IP12 | |
| | | | Avg | Max | Avg | Max | Avg | Max | Avg | Max | Avg | Max | Avg | Max |
| 15 | 4 | T | 14 | 53 | 4 | 22 | 4 | 22 | 21 | 70 | 1 | 6 | 2 | 15 |
| 15 | 4 | L | 3 | 23 | 1 | 2 | 0 | 1 | 13 | 47 | 0 | 1 | 0 | 1 |
| 15 | 6 | T | 16 | 54 | 9 | 40 | 6 | 40 | 26 | 91 | 0 | 1 | 3 | 28 |
| 15 | 6 | L | 1 | 2 | 1 | 2 | 0 | 1 | 13 | 47 | 0 | 1 | 0 | 1 |
| 20 | 4 | T | 22 | 58 | 4 | 9 | 4 | 9 | 23 | 58 | 2 | 9 | 2 | 9 |
| 20 | 4 | L | 10 | 24 | 6 | 12 | 4 | 12 | 12 | 50 | 5 | 18 | 5 | 18 |
| 20 | 6 | T | 35 | 86 | 3 | 15 | 3 | 12 | 35 | 86 | 6 | 25 | 6 | 25 |
| 20 | 6 | L | 10 | 40 | 1 | 3 | 1 | 3 | 16 | 45 | 3 | 31 | 0 | 1 |
| 25 | 4 | T | 12 | 50 | 2 | 4 | 3 | 15 | 17 | 69 | 1 | 6 | 2 | 6 |
| 25 | 4 | L | 7 | 11 | 7 | 12 | 4 | 5 | 14 | 41 | 2 | 7 | 3 | 6 |
| 25 | 6 | T | 48 | 105 | 9 | 30 | 9 | 30 | 45 | 92 | 10 | 33 | 10 | 33 |
| 25 | 6 | L | 15 | 35 | 8 | 28 | 7 | 28 | 29 | 70 | 11 | 34 | 12 | 34 |
| 30 | 4 | T | 17 | 52 | 4 | 9 | 3 | 8 | 18 | 52 | 5 | 16 | 6 | 16 |
| 30 | 4 | L | 26 | 71 | 7 | 19 | 7 | 29 | 24 | 66 | 4 | 14 | 5 | 15 |
| 30 | 6 | T | 26 | 61 | 7 | 16 | 7 | 16 | 29 | 76 | 6 | 24 | 6 | 24 |
| 30 | 6 | L | 39 | 99 | 18 | 29 | 15 | 29 | 27 | 52 | 10 | 33 | 9 | 27 |
| 40 | 4 | T | 16 | 40 | 6 | 13 | 6 | 13 | 13 | 31 | 3 | 6 | 3 | 6 |
| 40 | 4 | L | 16 | 33 | 11 | 19 | 6 | 12 | 18 | 40 | 4 | 23 | 5 | 23 |
| 40 | 6 | T | 26 | 45 | 11 | 35 | 9 | 35 | 29 | 74 | 12 | 34 | 12 | 34 |
| 40 | 6 | L | 34 | 56 | 24 | 46 | 17 | 36 | 31 | 63 | 5 | 23 | 6 | 23 |
| 50 | 4 | T | 12 | 23 | 8 | 27 | 5 | 10 | 13 | 20 | 3 | 9 | 3 | 9 |
| 50 | 4 | L | 14 | 22 | 11 | 26 | 6 | 17 | 11 | 19 | 3 | 9 | 3 | 6 |
| 50 | 6 | T | 23 | 57 | 18 | 30 | 16 | 39 | 29 | 55 | 14 | 31 | 6 | 13 |
| 50 | 6 | L | 37 | 57 | 26 | 39 | 23 | 44 | 37 | 52 | 11 | 34 | 13 | 33 |

The results in Table 52 present that the improvement procedures yield substantial gains in solution quality. The usage of the combination of both improvement procedures (IP12) consistently delivers the lowest deviations. Specifically, the Model B3–IP12-based heuristic achieves near-optimality in many difficult instances, reducing deviations to as low as 3% in loose setups where base models struggled. Furthermore, the gap between Model B2- and Model B3-based heuristics narrows significantly after improvement. While the Model B3–IP12-based heuristic

remains the most accurate, the Model B2–IP12-based heuristic often performs within a few percentage points of Model B3–IP12-based heuristic but at a fraction of the computational cost, reinforcing the utility of Model B2–IP12-based heuristic as an efficient middle ground.

Table 53: Frequency best and optimal solutions of Model B2- and B3-based heuristics after improvement procedures

| <i>n</i> | <i>m</i> | Setup | B2-W | | | | | | B3-W | | | | | |
|----------|----------|-------|------|----|-----|----|------|----|------|----|-----|----|------|----|
| | | | IP1 | | IP2 | | IP12 | | IP1 | | IP2 | | IP12 | |
| | | | FB | FO | FB | FO | FB | FO | FB | FO | FB | FO | FB | FO |
| 15 | 4 | T | 1 | 1 | 4 | 3 | 5 | 4 | 2 | 2 | 9 | 9 | 7 | 7 |
| 15 | 4 | L | 3 | 2 | 0 | 0 | 6 | 4 | 5 | 5 | 8 | 7 | 8 | 7 |
| 15 | 6 | T | 0 | 0 | 4 | 2 | 6 | 4 | 1 | 1 | 8 | 4 | 5 | 3 |
| 15 | 6 | L | 4 | 3 | 0 | 0 | 6 | 6 | 5 | 4 | 8 | 7 | 6 | 5 |
| 20 | 4 | T | 1 | 0 | 6 | 3 | 4 | 1 | 0 | 0 | 8 | 5 | 6 | 5 |
| 20 | 4 | L | 0 | 0 | 2 | 0 | 3 | 0 | 5 | 3 | 5 | 4 | 7 | 5 |
| 20 | 6 | T | 0 | 0 | 6 | 2 | 5 | 2 | 1 | 1 | 7 | 4 | 5 | 4 |
| 20 | 6 | L | 3 | 0 | 0 | 0 | 3 | 0 | 3 | 1 | 6 | 2 | 4 | 1 |
| 25 | 4 | T | 2 | 2 | 6 | 4 | 6 | 4 | 2 | 2 | 8 | 5 | 7 | 5 |
| 25 | 4 | L | 0 | 0 | 0 | 0 | 3 | 0 | 1 | 0 | 7 | 0 | 4 | 0 |
| 25 | 6 | T | 0 | 0 | 4 | 1 | 5 | 1 | 0 | 0 | 6 | 1 | 6 | 1 |
| 25 | 6 | L | 0 | 0 | 3 | 0 | 3 | 0 | 0 | 0 | 6 | 0 | 2 | 0 |
| 30 | 4 | T | 1 | 0 | 4 | 2 | 5 | 3 | 0 | 0 | 5 | 3 | 5 | 3 |
| 30 | 4 | L | 0 | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 7 | 1 | 4 | 0 |
| 30 | 6 | T | 1 | 1 | 4 | 1 | 4 | 1 | 1 | 0 | 7 | 1 | 5 | 1 |
| 30 | 6 | L | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 6 | 0 | 1 | 0 |
| 40 | 4 | T | 0 | 0 | 2 | 0 | 2 | 0 | 1 | 0 | 8 | 0 | 5 | 0 |
| 40 | 4 | L | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 8 | 1 | 3 | 0 |
| 40 | 6 | T | 0 | 0 | 4 | 1 | 6 | 1 | 0 | 0 | 5 | 1 | 3 | 1 |
| 40 | 6 | L | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 8 | 0 | 3 | 0 |
| 50 | 4 | T | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 8 | 1 | 5 | 0 |
| 50 | 4 | L | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 6 | 0 | 3 | 0 |
| 50 | 6 | T | 1 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 8 | 1 | 3 | 1 |
| 50 | 6 | L | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 9 | 0 | 2 | 0 |

In Table 53, the "Frequency Best" and "Frequency Optimal" analysis confirms the dominance of the combined approach. The Model B3–IP2- and B3–IP12-based heuristics act as the leading methods, achieving the best-found solution (FB) in the highest number of instances across all categories. However, for smaller or tighter instances, the Model B2–IP12-based heuristic frequently matches or even exceeds the performance of the B3 variant. For instance, in the setup $n = 15, m = 6$, tight, Model B2–IP12-based heuristic finds the best solution 6 times, whereas Model B3–IP12-based heuristic finds it only 5 times. This indicates that for less complex

problems, the aggressive search of the Model B3–IP12-based heuristic is unnecessary, and the lighter Model B2–IP12-based heuristic suffices to find the best available solution.

Overall, the results reveal a clear performance hierarchy among the block formation models and improvement procedures. Models B0 and B1 are extremely fast but yield poor solution quality, and they remain weak even after the application of improvement procedures. In contrast, B2 provides a strong balance between computational efficiency and solution quality, while B3 achieves the best overall performance, particularly as instance size increases. Both B2 and B3 benefit substantially from the improvement mechanisms. Among these, IP1 delivers the smallest quality gains, whereas IP12 consistently attains the lowest average deviations, although IP2 occasionally outperforms IP12. Based on these findings, the main experiments proceed with block formation models B2 and B3 and with improvement procedures IP2 and IP12.

6.4 Computational Experiments

To evaluate the performance of the proposed approaches, extensive computational experiments are conducted for the mathematical models, the B&B algorithm, and the heuristic methods. The objective of this section is not only to compare computational performance but also to understand how some problem characteristics and parameters influence the solution time and solution quality.

6.4.1 Mathematical Models

We first analyze the behavior of the stochastic programming (SP) model under different structural configurations. Results are reported in Table 54. For instances that cannot be solved within the time limit, the termination time of 7200 seconds is included in the average CPU time to provide a realistic measure of computational burden.

We use the following setup:

- Number of operations: 15
- Number of workstations: 4 and 6
- Number of operations per block: 2 and 4
- Number of blocks per workstation: 2 and 4
- Maintenance type: Case 1 and Case 2
- Maintenance probabilities: Identical probabilities
- Precedence structure: Tight and loose
- Operation time distributions: U[10, 20]
- Earliest and latest workstation parameters (E_i and L_i): Included

Table 54: The results for the experiment

| m | n₀ | m₀ | Case Type | Setup | SP Avg. CPU Time | SP Max. CPU Time | VSS Avg |
|----------|----------------------|----------------------|------------------|--------------|-------------------------|-------------------------|----------------|
| 4 | 2 | 2 | 1 | T | 1.71 | 13.34 | 1.09 |
| 4 | 2 | 2 | 1 | L | 8.75 | 62.42 | 0.76 |
| 4 | 2 | 2 | 2 | T | 3.09 | 22.25 | 0.36 |
| 4 | 2 | 2 | 2 | L | 15.11 | 98.00 | 0.18 |
| 4 | 2 | 4 | 1 | T | 2.96 | 10.13 | 1.91 |
| 4 | 2 | 4 | 1 | L | 2.39 | 8.36 | 2.61 |
| 4 | 2 | 4 | 2 | T | 2.52 | 4.36 | 25.87 |
| 4 | 2 | 4 | 2 | L | 2.79 | 4.41 | 31.31 |
| 4 | 4 | 2 | 1 | T | 1.63 | 12.80 | 1.07 |
| 4 | 4 | 2 | 1 | L | 6.80 | 44.08 | 0.81 |
| 4 | 4 | 2 | 2 | T | 2.85 | 21.69 | 0.15 |
| 4 | 4 | 2 | 2 | L | 14.55 | 90.92 | 0.22 |
| 4 | 4 | 4 | 1 | T | 0.67 | 2.69 | 1.95 |
| 4 | 4 | 4 | 1 | L | 2.19 | 4.92 | 3.51 |
| 4 | 4 | 4 | 2 | T | 1.30 | 3.02 | 22.31 |
| 4 | 4 | 4 | 2 | L | 0.81 | 1.63 | 29.64 |
| 6 | 2 | 2 | 1 | T | 299.52 | 2820.80 | 1.50 |
| 6 | 2 | 2 | 1 | L | 1263.49 | 7200.00 | 0.81 |
| 6 | 2 | 2 | 2 | T | 330.98 | 2629.30 | 11.10 |
| 6 | 2 | 2 | 2 | L | 1359.52 | 7200.00 | 13.00 |
| 6 | 2 | 4 | 1 | T | 193.49 | 1456.70 | 4.39 |
| 6 | 2 | 4 | 1 | L | 272.49 | 2057.81 | 4.18 |
| 6 | 2 | 4 | 2 | T | 69.81 | 152.84 | 45.42 |
| 6 | 2 | 4 | 2 | L | 52.95 | 102.44 | 59.70 |
| 6 | 4 | 2 | 1 | T | 329.33 | 3095.20 | 1.83 |
| 6 | 4 | 2 | 1 | L | 1150.70 | 7200.00 | 2.10 |
| 6 | 4 | 2 | 2 | T | 333.35 | 2657.80 | 15.78 |
| 6 | 4 | 2 | 2 | L | 1403.96 | 7200.00 | 17.26 |
| 6 | 4 | 4 | 1 | T | 57.02 | 410.88 | 3.30 |
| 6 | 4 | 4 | 1 | L | 240.41 | 742.94 | 6.05 |

| m | n₀ | m₀ | Case Type | Setup | SP Avg. CPU Time | SP Max. CPU Time | VSS Avg |
|----------|----------------------|----------------------|------------------|--------------|-------------------------|-------------------------|----------------|
| 6 | 4 | 4 | 2 | T | 23.96 | 36.61 | 40.40 |
| 6 | 4 | 4 | 2 | L | 17.65 | 33.02 | 53.56 |

We next investigate the effect of the number of the workstations on the complexity of the solutions. The results clearly show that the computational performance of the mathematical model is highly sensitive to the number of workstations. This behavior is expected as increasing the number of workstations expands the assignment possibilities for each operation and weakens implicit capacity restrictions. Consequently, the feasible solution space grows rapidly, which significantly increases the complexity of the search process. For every configuration, increasing the number of workstations from 4 to 6 leads to a substantial increase in both average and maximum CPU times.

We also look for the effect of the maintenance structure and observe that the maintenance structure plays a critical role in computational tractability. In most instances, the mathematical model is easier to solve under "no breakdown" or "minor breakdown" conditions (Case 1) compared to scenarios that include "major breakdowns" (Case 2). The introduction of severe disruptions increases uncertainty in workload allocations, which in turn increases the complexity of balancing operations across workstations. This result is intuitive from a modeling perspective: higher variability obscures the problem's underlying constraints, which reduces the efficiency of optimization pruning mechanisms.

Table 54 also reports on the Value of the Stochastic Solution (VSS) results. Regarding the value of incorporating uncertainty, the results in Table 54 indicate that the VSS remains consistently high for Case 2 across nearly all configurations, confirming the practical necessity of the stochastic formulation. As expected, Case 2 represents more severe breakdown scenarios, and the VSS values are generally higher than those observed in Case 1. This pattern indicates that although the stochastic model provides benefits in both maintenance settings, its relative

advantage over the deterministic expected value solution becomes substantially more pronounced in environments characterized by major and frequent disruptions.

We analyze the performance of the SP and CP models under the following problem configurations: $n = 15, 20, 25, 30$, $m = 4, 6$, $n_o = 4$, $m_o = 4$, identical scenario probabilities, and Case 1 maintenance type. The computational results are reported in Table 55.

Table 55: SP and CP model results

| n | m | Setup | SP Avg. CPU Time | SP Max. CPU Time | CP Avg. CPU Time | CP Max. CPU Time |
|-----|-----|-------|------------------|------------------|------------------|------------------|
| 15 | 4 | T | 0.67 | 2.69 | 8.92 | 46.57 |
| 15 | 4 | L | 2.19 | 4.92 | 73.27 | 198.78 |
| 15 | 6 | T | 57.02 | 410.88 | 891.05 | 7200.00 |
| 15 | 6 | L | 240.41 | 742.94 | 2717.48 | 7200.00 |
| 20 | 4 | T | 5.93 | 23.34 | 260.67 | 1280.93 |
| 20 | 4 | L | 1032.31 | 7200.00 | 7200.00 | 7200.00 |
| 20 | 6 | T | 142.87 | 1213.76 | 2051.86 | 7200.00 |
| 20 | 6 | L | 1372.44 | 7200.00 | 5807.49 | 7200.00 |
| 25 | 4 | T | 58.02 | 259.29 | 67.55 | 313.43 |
| 25 | 4 | L | 3418.84 | 7200.00 | 6201.87 | 7200.00 |
| 25 | 6 | T | 826.04 | 4860.70 | 1089.50 | 7200.00 |
| 25 | 6 | L | 7200.00 | 7200.00 | 7200.00 | 7200.00 |
| 30 | 4 | T | 83.23 | 211.00 | 101.10 | 656.63 |
| 30 | 4 | L | 5207.41 | 7200.00 | 7200.00 | 7200.00 |
| 30 | 6 | T | 1286.47 | 7200.00 | 1816.90 | 7200.00 |
| 30 | 6 | L | 7200.00 | 7200.00 | 7200.00 | 7200.00 |

The results indicate that the SP model consistently demonstrates superior computational efficiency compared to the CP model across all tested configurations. Nevertheless, both exact approaches exhibit clear computational limitations. The CP model begins to encounter these difficulties even for relatively small-sized instances; for example, in loose configurations with 6 workstations and 20 operations, the CP model reaches the termination time limit, whereas the SP model remains solvable with an average computation time of 1032.11 seconds. As problem size increases, the performance of the SP model also deteriorates, particularly under loose constraint structures. When the number of operations approaches 25, average CPU times frequently reach the termination time limit, indicating a substantial increase in computational complexity.

To further investigate the performance of the SP model, additional experiments are conducted for $n = 40$ and 50 . The corresponding results are presented in Table 56.

Table 56: SP Model Results

| n | m | Setup | SP Avg. CPU Time | SP Max. CPU Time |
|-----|-----|-------|------------------|------------------|
| 40 | 4 | T | 3579.62 | 7200.00 |
| 40 | 4 | L | 7200.00 | 7200.00 |
| 40 | 6 | T | 5758.56 | 7200.00 |
| 40 | 6 | L | 7200.00 | 7200.00 |
| 50 | 4 | T | 6607.10 | 7200.00 |
| 50 | 4 | L | 7200.00 | 7200.00 |
| 50 | 6 | T | 6803.96 | 7200.00 |
| 50 | 6 | L | 7200.00 | 7200.00 |

These results indicate that the SP model approaches its practical computational limit at $n = 50$, where most instances cannot be solved within the time limit. This observation confirms that although the SP formulation remains effective for small and medium-sized instances, its applicability to large-sized problems is inherently constrained by computational complexity.

To evaluate solution quality when optimality cannot be proven within the time limit, the optimality gap percentages of the SP model are analyzed and reported in Table 57. These values quantify the distance between the best feasible solution obtained and the best available bound found by the solver at the termination time.

Table 57: Percent Optimality Gap of SP model for 15min and 2 hours

| n | m | Setup | Avg. of SP GAP (15min) | Max. of SP GAP (15min) | Avg. of SP GAP (2hour) | Max. of SP GAP (2hour) |
|-----|-----|-------|------------------------|------------------------|------------------------|------------------------|
| 15 | 4 | T | 0.00 | 0.01 | 0.00 | 0.01 |
| 15 | 4 | L | 0.01 | 0.01 | 0.01 | 0.01 |
| 15 | 6 | T | 0.01 | 0.01 | 0.01 | 0.01 |
| 15 | 6 | L | 0.01 | 0.01 | 0.01 | 0.01 |
| 20 | 4 | T | 0.01 | 0.01 | 0.01 | 0.01 |
| 20 | 4 | L | 1.39 | 13.82 | 0.64 | 6.34 |
| 20 | 6 | T | 0.25 | 2.49 | 0.01 | 0.01 |
| 20 | 6 | L | 0.64 | 4.44 | 0.20 | 1.91 |
| 25 | 4 | T | 0.00 | 0.01 | 0.00 | 0.01 |
| 25 | 4 | L | 5.81 | 21.89 | 2.67 | 16.14 |
| 25 | 6 | T | 0.97 | 9.59 | 0.01 | 0.01 |
| 25 | 6 | L | 9.99 | 16.13 | 6.65 | 14.41 |
| 30 | 4 | T | 0.01 | 0.01 | 0.01 | 0.01 |
| 30 | 4 | L | 5.37 | 13.15 | 2.63 | 8.42 |

| <i>n</i> | <i>m</i> | Setup | Avg. of SP GAP (15min) | Max. of SP GAP (15min) | Avg. of SP GAP (2hour) | Max. of SP GAP (2hour) |
|----------|----------|-------|------------------------|------------------------|------------------------|------------------------|
| 30 | 6 | T | 1.32 | 9.59 | 0.52 | 5.13 |
| 30 | 6 | L | 19.55 | 23.65 | 18.33 | 23.15 |
| 40 | 4 | T | 2.11 | 13.46 | 1.03 | 8.03 |
| 40 | 4 | L | 16.35 | 26.32 | 14.30 | 23.20 |
| 40 | 6 | T | 1.58 | 6.59 | 1.15 | 4.71 |
| 40 | 6 | L | 19.05 | 30.88 | 16.26 | 28.38 |
| 50 | 4 | T | 9.32 | 24.18 | 7.61 | 21.74 |
| 50 | 4 | L | 19.70 | 32.56 | 15.63 | 31.40 |
| 50 | 6 | T | 8.40 | 19.14 | 6.06 | 13.91 |
| 50 | 6 | L | 19.72 | 34.06 | 14.09 | 29.50 |
| 75 | 10 | T | 90.53 | 95.66 | 86.82 | 93.94 |
| 75 | 10 | L | 94.32 | 98.55 | 91.86 | 94.87 |
| 100 | 10 | T | 94.16 | 96.88 | 90.96 | 93.44 |
| 100 | 10 | L | 98.55 | 99.75 | 96.09 | 98.98 |

The optimality gap results reveal structural patterns consistent with the computational trends observed earlier. Increasing the number of workstations increases the difficulty of the problem. This effect stems from the combinatorial expansion introduced by partitioning operations across a larger number of workstations, which weakens bounding effectiveness and limits the solver’s ability to close the optimality gap.

Across all problem sizes with more than 15 operations, loose instances exhibit substantially higher optimality gaps than tight instances. This confirms that higher constraint density strengthens the formulation by reducing the feasible region, whereas loose precedence relations expand the search space beyond the solver’s capacity.

The performance of the SP model begins to degrade noticeably for $n = 40, 50$, where average optimality gaps for loose instances consistently exceed 15–19%. For larger problem sizes with 75 and 100 operations, the model becomes practically ineffective, with average optimality gaps exceeding 90%, indicating that optimality cannot be meaningfully approximated within the time limit.

The magnitude of the optimality gap is also critical for interpreting heuristic results. When the optimality gap is below a specified threshold (e.g., 2%),

deviations of heuristic solutions from the SP solution are reported directly. However, when the optimality gap exceeds this threshold, heuristic methods are evaluated based on their relative performance with respect to one another.

6.4.2 B&B Algorithm

The performance of the Branch and Bound (B&B) algorithm under different parameters is reported in Table 58. The table reports the average and maximum CPU times and deviations from the optimal or best-known solutions where the deviations are defined as:

$$\text{Percent Deviation} = 100 * (z^*_{BAB} - z^*_{SP}) / z^*_{SP}$$

Table 58: B&B Results; CPU times and percent deviations

| <i>n</i> | <i>m</i> | Setup | Avg. CPU Time | Max. CPU Time | Avg. Deviation from Optimal | Max. Deviation from Optimal |
|----------|----------|-------|---------------|---------------|-----------------------------|-----------------------------|
| 15 | 4 | T | 4772.76 | 7200.00 | 6 | 32 |
| 15 | 4 | L | 7200.00 | 7200.00 | 31 | 48 |
| 15 | 6 | T | 6426.61 | 7200.00 | 5 | 23 |
| 15 | 6 | L | 7200.00 | 7200.00 | 3 | 17 |
| 20 | 4 | T | 7200.00 | 7200.00 | 9 | 25 |
| 20 | 4 | L | 7200.00 | 7200.00 | 35 | 100 |
| 20 | 6 | T | 7200.00 | 7200.00 | 35 | 125 |
| 20 | 6 | L | 7200.00 | 7200.00 | 67 | 126 |
| 25 | 4 | T | 7200.00 | 7200.00 | 17 | 44 |
| 25 | 4 | L | 7200.00 | 7200.00 | 26 | 61 |
| 25 | 6 | T | 7179.07 | 7200.00 | 21 | 76 |
| 25 | 6 | L | 7200.00 | 7200.00 | 46 | 107 |

The results demonstrate that the performance is influenced by the number of operations, the number of workstations, and the constraint structure of the instances.

Consistent with the behavior of the mathematical models, the number of workstations has a substantial impact on the performance of the B&B algorithm. This impact becomes particularly pronounced when the number of operations reaches 20 or more. As the number of workstations increases from 4 to 6, the deviation increases notably; for example, when $n = 20$, the average deviation rises

from 35% to 67%. This deterioration reflects the rapid expansion of the solution space, which requires the exploration of a significantly larger branching tree.

The results also demonstrate that loose instances are considerably more difficult to solve than tight instances. For example, in configurations with 15 operations and 4 workstations, loose instances consistently reach the termination limit, whereas tight instances achieve optimality in several cases. This difficulty is further reflected in solution quality: instances reaching the time limit exhibit average deviations ranging from 3% to 67%, while tight instances maintain a narrower deviation range between 5% and 35%.

Examining the interaction between the number of workstations and precedence structure reveals specific limitations in the algorithm's consistency. When the number of workstations increases to 6, deviations from optimality become severe for loose instances, reaching average deviations of 67% and 46% for 20 and 25 operations, respectively, with maximum deviations exceeding 100%. This suggests that the B&B algorithm struggles to prove optimality in loose instances due to the sheer magnitude of the solution space, despite the use of bounding mechanisms. Conversely, increasing the number of workstations appears to enhance the effectiveness of bounding mechanisms for tight instances, keeping deviations within the 21% to 35% range for 20 and 25 operations. However, as the number of operations increases to 20, both instance types fail to achieve proven optimality within the time limit, highlighting the necessity for heuristic approaches to handle large-sized problems.

6.4.3 Heuristic Approaches

6.4.3.1 B&B-based Heuristic

The performance of the B&B-based heuristic is analyzed and summarized in Table 59, where it is directly compared with the B&B algorithm in terms of computational time and deviation from optimality.

Table 59: B&B and B&B-based heuristic results; CPU times and percent deviations

| <i>n</i> | <i>m</i> | Setu p | B&B CPU Time Avg. | B&B Max. CPU Time | B&B Avg. Dev. from Optimal | B&B- Based CPU Time Avg. | B&B- Based Max. CPU Time | B&B- Based Avg. Dev. from Optimal |
|----------|----------|-----------|----------------------------|----------------------------|--|-----------------------------------|-----------------------------------|--|
| 15 | 4 | T | 4772.76 | 7200.00 | 6 | 1,429.14 | 4,326.45 | 9 |
| 15 | 4 | L | 7200.00 | 7200.00 | 31 | 5,274.04 | 7,200.00 | 32 |
| 15 | 6 | T | 6426.61 | 7200.00 | 5 | 1,553.02 | 7,200.00 | 8 |
| 15 | 6 | L | 7200.00 | 7200.00 | 3 | 1,897.09 | 7,200.00 | 4 |
| 20 | 4 | T | 7200.00 | 7200.00 | 9 | 4,300.26 | 7,200.00 | 13 |
| 20 | 4 | L | 7200.00 | 7200.00 | 35 | 6,715.84 | 7,200.00 | 35 |
| 20 | 6 | T | 7200.00 | 7200.00 | 35 | 6,412.43 | 7,200.00 | 37 |
| 20 | 6 | L | 7200.00 | 7200.00 | 67 | 7,200.00 | 7,200.00 | 79 |
| 25 | 4 | T | 7200.00 | 7200.00 | 17 | 5,105.62 | 7,200.00 | 19 |
| 25 | 4 | L | 7200.00 | 7200.00 | 26 | 7,200.00 | 7,200.00 | 27 |
| 25 | 6 | T | 7179.07 | 7200.00 | 21 | 5,781.79 | 7,200.00 | 22 |
| 25 | 6 | L | 7200.00 | 7200.00 | 46 | 7,200.00 | 7,200.00 | 66 |

For instances with 15 operations, the B&B-based heuristic maintains competitive solution quality across all configurations, with several instances exhibiting deviations below 10%. As expected, the deviations are marginally higher than those obtained by the B&B algorithm. However, this slight loss in solution quality is accompanied by a notable improvement in computational performance. The B&B-based heuristic reaches feasible solutions in shorter times and exhibits fewer cases that terminate at the time limit. These results confirm the expected trade-off between solution quality and computational effort.

As the number of operations increases, both algorithms frequently encounter instances that cannot be solved to proven optimality within the time limit. Nevertheless, even in such cases, the B&B-based heuristic often identifies feasible solutions earlier than the B&B algorithm, particularly for tighter configurations. This behavior indicates that the heuristic guidance effectively accelerates the search process without significantly compromising solution quality.

6.4.3.2 Mathematical Model- and Rule-based Heuristic

Based on the findings of the preliminary experiments, Model B2- and B3-based heuristics are identified as the most effective block formation models. We first report the number of blocks generated by these models and the number of block sets formed under the workstation inclusion constraints. Table 60 presents the average number of blocks, the average number of block sets, and the corresponding CPU times for both models.

Table 60: The number of blocks and block sets and CPU times by B2 and B3

| n | m | Setup | B2-W | | | B3-W | | |
|-----|-----|-------|-----------------------|---------------------------|----------------|-----------------------|---------------------------|----------------|
| | | | Avg. number of blocks | Avg. number of block-sets | Avg. CPU Times | Avg. number of blocks | Avg. number of block-sets | Avg. CPU Times |
| 15 | 4 | T | 5 | 4 | 0.12 | 5 | 4 | 1.39 |
| 15 | 4 | L | 4 | 4 | 0.02 | 4 | 3 | 4.56 |
| 15 | 6 | T | 5 | 4 | 0.54 | 5 | 4 | 12.78 |
| 15 | 6 | L | 4 | 4 | 0.09 | 4 | 3 | 41.58 |
| 20 | 4 | T | 6 | 4 | 0.16 | 6 | 5 | 7.19 |
| 20 | 4 | L | 5 | 5 | 0.03 | 5 | 4 | 176.25 |
| 20 | 6 | T | 6 | 5 | 1.57 | 6 | 5 | 102.76 |
| 20 | 6 | L | 5 | 5 | 0.10 | 5 | 4 | 603.13 |
| 25 | 4 | T | 8 | 5 | 0.14 | 8 | 5 | 6.90 |
| 25 | 4 | L | 7 | 6 | 0.12 | 7 | 5 | 757.80 |
| 25 | 6 | T | 8 | 6 | 2.23 | 8 | 5 | 235.90 |
| 25 | 6 | L | 7 | 6 | 0.12 | 7 | 5 | 900.03 |
| 30 | 4 | T | 9 | 5 | 0.18 | 9 | 5 | 60.68 |
| 30 | 4 | L | 8 | 6 | 0.08 | 8 | 5 | 900.00 |
| 30 | 6 | T | 9 | 6 | 4.86 | 9 | 6 | 447.50 |
| 30 | 6 | L | 8 | 6 | 0.14 | 8 | 6 | 900.04 |
| 40 | 4 | T | 10 | 6 | 0.14 | 10 | 6 | 498.51 |
| 40 | 4 | L | 10 | 6 | 0.08 | 10 | 6 | 900.00 |
| 40 | 6 | T | 10 | 6 | 10.19 | 10 | 6 | 900.02 |
| 40 | 6 | L | 10 | 7 | 0.25 | 10 | 6 | 900.04 |
| 50 | 4 | T | 13 | 6 | 0.11 | 13 | 7 | 900.00 |
| 50 | 4 | L | 13 | 7 | 0.09 | 13 | 6 | 900.01 |
| 50 | 6 | T | 13 | 7 | 0.42 | 13 | 7 | 900.04 |
| 50 | 6 | L | 13 | 7 | 0.39 | 13 | 7 | 900.04 |
| 75 | 10 | T | 19 | 16 | 22.30 | 21 | 17 | 900.00 |
| 75 | 10 | L | 19 | 15 | 2.61 | 19 | 15 | 900.00 |
| 100 | 10 | T | 25 | 18 | 170.53 | 36 | 28 | 900.00 |
| 100 | 10 | L | 25 | 19 | 4.73 | 27 | 20 | 900.00 |

The computational results reveal a distinct performance trade-off between the Model B2- and B3-based heuristics. Model B2-based heuristic demonstrates

superior computational efficiency, maintaining low average CPU times for the majority of instances. This disparity is reflected in the complexity of the solutions; while both variants generate a similar number of blocks and block sets for smaller problems, Model B3-based heuristic explores a denser solution space in larger, tighter instances. For the largest tested configuration, Model B3-based heuristic generates an average of 36 blocks and 28 block sets compared to 25 blocks and 18 block sets for Model B2-based heuristic, directly correlating with its significantly higher computational burden. Consequently, Model B2-based heuristic emerges as the preferred choice for rapid solution generation in loose constraints, while Model B3-based heuristic exhausts the time limit to maximize search depth in complex scenarios.

Based on these results and the results of the preliminary experiments, the main computational experiments proceed with block formation models B2 and B3 using block-sets and improvement procedures IP2 and IP12.

Two main metaheuristic, “Best of 2” and “Best of 4”, approaches are proposed for the mathematical model-based heuristics:

- “Best of 2”; consists of:
 - (i) the Model B2-based heuristic
 - (ii) the Model B3-based heuristic
- “Best of 4” consists of:
 - (i) the Model B2–IP2-based heuristic
 - (ii) the Model B3–IP2-based heuristic
 - (iii) the Model B2–IP12-based heuristic
 - (iv) the Model B3–IP12-based heuristic

To evaluate computational performance, Table 61 reports the average and maximum CPU times of the “Best of 2”, “Best of 4”, and Rule-based heuristics.

CPU times represent the cumulative time of all executed models and improvement procedures.

Table 61: CPU Times of matheuristics and Rule-based heuristic

| <i>n</i> | <i>m</i> | Setup | Best of 2 | | Best of 4 | | Rule-Based Heuristic | | |
|----------|----------|-------|-----------|---------|-----------|---------|----------------------|--------|------------|
| | | | Avg | Max | Avg | Max | Avg | Max | Freq. Feas |
| 15 | 4 | T | 1.51 | 5.97 | 2.95 | 8.15 | 0.22 | 0.38 | 9 |
| 15 | 4 | L | 4.58 | 7.58 | 6.07 | 8.98 | 0.18 | 0.26 | 10 |
| 15 | 6 | T | 13.32 | 67.53 | 21.41 | 116.07 | 0.35 | 0.64 | 9 |
| 15 | 6 | L | 41.67 | 97.91 | 48.89 | 101.60 | 0.71 | 1.58 | 10 |
| 20 | 4 | T | 7.36 | 29.31 | 10.92 | 32.08 | 0.35 | 0.77 | 7 |
| 20 | 4 | L | 176.28 | 900.06 | 210.03 | 902.07 | 0.48 | 0.72 | 10 |
| 20 | 6 | T | 104.32 | 527.67 | 109.22 | 536.05 | 0.42 | 0.57 | 7 |
| 20 | 6 | L | 603.23 | 900.22 | 608.49 | 909.32 | 0.45 | 0.92 | 10 |
| 25 | 4 | T | 7.05 | 36.59 | 8.83 | 40.74 | 0.25 | 0.29 | 2 |
| 25 | 4 | L | 757.92 | 900.78 | 769.20 | 947.94 | 1.03 | 3.80 | 9 |
| 25 | 6 | T | 238.14 | 900.22 | 242.20 | 902.21 | 0.85 | 1.23 | 2 |
| 25 | 6 | L | 900.15 | 900.22 | 907.75 | 923.45 | 0.70 | 1.61 | 9 |
| 30 | 4 | T | 60.86 | 538.80 | 63.36 | 550.45 | infeas | infeas | 0 |
| 30 | 4 | L | 900.09 | 900.23 | 909.10 | 933.40 | 1.43 | 4.45 | 8 |
| 30 | 6 | T | 452.36 | 900.38 | 456.81 | 905.73 | infeas | infeas | 0 |
| 30 | 6 | L | 900.18 | 900.33 | 1000.28 | 1804.14 | 1.13 | 2.47 | 8 |
| 40 | 4 | T | 498.65 | 900.14 | 503.29 | 918.96 | 0.52 | 0.82 | 3 |
| 40 | 4 | L | 900.09 | 900.14 | 1095.34 | 1838.07 | 26.20 | 78.91 | 8 |
| 40 | 6 | T | 910.21 | 998.66 | 916.01 | 1003.01 | 0.75 | 0.94 | 3 |
| 40 | 6 | L | 900.29 | 900.44 | 943.82 | 1054.92 | 0.77 | 1.51 | 7 |
| 50 | 4 | T | 900.12 | 900.16 | 1293.99 | 2536.11 | infeas | infeas | 0 |
| 50 | 4 | L | 900.10 | 900.11 | 2237.72 | 4501.89 | infeas | infeas | 0 |
| 50 | 6 | T | 900.46 | 900.98 | 940.67 | 1086.42 | infeas | infeas | 0 |
| 50 | 6 | L | 900.43 | 900.58 | 1019.25 | 1812.20 | infeas | infeas | 0 |
| 75 | 10 | T | 953.79 | 1028.66 | 1406.68 | 1747.01 | 0.01 | 0.03 | 6 |
| 75 | 10 | L | 1142.51 | 2701.97 | 1958.80 | 4291.22 | 0.00 | 0.01 | 6 |
| 100 | 10 | T | 1386.48 | 1961.03 | 2422.22 | 4036.51 | 0.01 | 0.02 | 5 |
| 100 | 10 | L | 1337.42 | 2703.22 | 3437.66 | 6498.50 | 0.01 | 0.01 | 4 |

The “Best of 2” approach, which selects the better solution between the Model B2– and Model B3–based heuristics, exhibits moderate computational requirements. Although its CPU times are significantly higher than those of the Rule-based heuristic, they remain within reasonable limits and show a stable growth pattern as the problem size increases. In contrast, the “Best of 4” approach is clearly the most computationally demanding among the heuristics. Since this approach evaluates four different variants, it naturally incurs higher average and maximum CPU times. This indicates that the “Best of 4” strategy trades computational efficiency for the

potential of improved solution quality, particularly for tightly constrained instances.

The Rule-based heuristic demonstrates negligible CPU times in all instances, with average times consistently close to zero. However, this speed advantage comes at the severe expense of solution feasibility. The number of feasible solutions drops drastically for complex problem settings; notably, for $n = 50$, the feasible rate drops to zero for all cases, and for $n = 100$ it finds a feasible solution in only 4 and 5 out of 10 instances. These results highlight the limitations of purely rule-driven constructive procedures for large-sized problem instances.

The reliability of the Rule-based heuristic is further supported by its infeasibility detection mechanism. The experimental results confirm that infeasible configurations are consistently identified by the predefined logical conditions. In all cases where a feasible solution could not be constructed due to conflicting constraints, the mechanism correctly detected infeasibility.

In summary, the computational results reveal a clear trade-off between solution quality and computational effort. While Rule-based heuristic is extremely fast, they fail to provide feasible solutions for large-sized problem instances. In contrast, the “Best of 2” and particularly the “Best of 4” strategies require greater computational effort but provide reliable and high-quality solutions, making them necessary for complex industrial systems.

We next analyze the results of the “Best of 4”, “Best of 2” matheuristics’ and Rule-based heuristic’s deviations from SP model solutions and report the results in Table 62.

The deviations are defined as the difference between the objective function values of the heuristic solution and optimal solution as a ratio of the optimal solution.

Table 62: Percent deviations of matheuristics and Rule-based heuristic

| n | m | Setup | Best of 2 | | Best of 4 | | | Rule-Based Heuristic | | |
|-----|-----|-------|-----------|-------|-----------|-------|------------|----------------------|--------|------------|
| | | | Avg | Max | Avg | Max | Freq. Opt* | Avg | Max | Freq. Feas |
| 15 | 4 | T | 12.93 | 49.55 | 0.17 | 1.68 | 9 | 59.29 | 105.27 | 9 |
| 15 | 4 | L | 4.09 | 28.71 | 0.11 | 0.75 | 8 | 38.06 | 86.57 | 10 |
| 15 | 6 | T | 20.04 | 96.18 | 0.29 | 1.32 | 5 | 73.37 | 108.36 | 9 |
| 15 | 6 | L | 1.53 | 3.56 | 0.02 | 0.16 | 9 | 38.19 | 86.57 | 10 |
| 20 | 4 | T | 21.51 | 58.11 | 1.86 | 8.86 | 6 | 50.21 | 99.71 | 7 |
| 20 | 4 | L | 10.29 | 27.44 | 2.11 | 8.47 | 5 | 52.67 | 89.74 | 10 |
| 20 | 6 | T | 35.60 | 85.72 | 1.23 | 9.54 | 5 | 90.42 | 143.40 | 7 |
| 20 | 6 | L | 7.02 | 45.42 | 0.19 | 0.55 | 3 | 75.84 | 127.78 | 10 |
| 25 | 4 | T | 8.07 | 32.32 | 0.57 | 3.26 | 6 | 14.42 | 20.25 | 2 |
| 25 | 4 | L | 13.23 | 31.11 | 1.84 | 3.96 | 0 | 45.89 | 71.69 | 9 |
| 25 | 6 | T | 39.25 | 68.34 | 7.64 | 30.26 | 1 | 49.33 | 61.64 | 2 |
| 25 | 6 | L | 27.14 | 39.49 | 6.00 | 28.24 | 1 | 87.31 | 128.53 | 9 |
| 30 | 4 | T | 15.72 | 51.52 | 2.10 | 7.06 | 5 | infeas | infeas | 0 |
| 30 | 4 | L | 19.86 | 41.75 | 2.74 | 8.66 | 2 | 44.16 | 73.76 | 8 |
| 30 | 6 | T | 22.95 | 68.66 | 2.49 | 11.40 | 1 | infeas | infeas | 0 |
| 30 | 6 | L | 34.09 | 95.88 | 5.81 | 12.73 | 0 | 71.64 | 106.70 | 8 |
| 40 | 4 | T | 13.40 | 38.12 | 2.46 | 6.42 | 0 | 35.00 | 44.48 | 3 |
| 40 | 4 | L | 14.78 | 33.53 | 2.70 | 9.82 | 1 | 36.24 | 42.32 | 8 |
| 40 | 6 | T | 22.70 | 45.25 | 8.49 | 34.22 | 1 | 74.28 | 86.86 | 3 |
| 40 | 6 | L | 31.98 | 67.21 | 3.69 | 8.15 | 0 | 73.69 | 82.57 | 7 |
| 50 | 4 | T | 13.18 | 20.89 | 2.58 | 8.86 | 1 | infeas | infeas | 0 |
| 50 | 4 | L | 10.00 | 19.87 | 2.37 | 4.52 | 0 | infeas | infeas | 0 |
| 50 | 6 | T | 28.28 | 53.23 | 5.25 | 13.44 | 2 | infeas | infeas | 0 |
| 50 | 6 | L | 31.30 | 45.83 | 10.69 | 32.96 | 0 | infeas | infeas | 0 |

The results in Table 62 demonstrate a clear hierarchy in solution quality among the heuristics. The "Best of 4" matheuristic consistently outperforms the "Best of 2" matheuristic across all problem configurations, achieving significantly lower average deviations from the best-known solution. This trend holds true for larger instances as well. This confirms that the additional computational effort invested in Improvement Procedures 1 and 2 (integrated into the Best of 4 strategy) yields substantial returns in solution accuracy. In contrast, the Rule-based heuristic proves to be unreliable for this class of problems. Its average deviations are excessively high, frequently exceeding 50%. Furthermore, the Rule-based heuristic fails to generate feasible solutions entirely for highly constrained instances, for all setups with $n = 50$. This inability to navigate the complex inclusion/exclusion constraints of larger transfer lines validates the necessity of the proposed model-based heuristics over simple constructive rules.

Finally, the frequency optimal is measured by counting same results with SP best-known solution and the frequency optimal column highlights the robustness of the "Best of 4" matheuristic in finding optimal or best-known solutions. For smaller instances, it identifies the optimal solution in nearly all cases. Although this frequency naturally decreases as problem complexity grows, the "Best of 4" matheuristic remains the most viable option, consistently keeping maximum deviations well below those of its counterparts.

For instances where the SP model achieves near-optimality (optimality gap < 2%), deviations from the SP solution are reported in Table 63.

Table 63: Deviations from SP solution when optimality gap < 2%

| <i>n</i> | <i>m</i> | Setup | Best of 2 | | Best of 4 | | | Rule-Based | | |
|----------|----------|-------|-----------|-------|-----------|-------|------------|------------|--------|------------|
| | | | Avg | Max | Avg | Max | Freq. Opt* | Avg | Max | Freq. Feas |
| 15 | 4 | T | 12.93 | 49.55 | 0.17 | 1.68 | 9 | 59.29 | 105.27 | 9 |
| 15 | 4 | L | 4.09 | 28.71 | 0.11 | 0.75 | 8 | 38.06 | 86.57 | 10 |
| 15 | 6 | T | 20.04 | 96.18 | 0.29 | 1.32 | 5 | 73.37 | 108.36 | 9 |
| 15 | 6 | L | 1.53 | 3.56 | 0.02 | 0.16 | 9 | 38.19 | 86.57 | 10 |
| 20 | 4 | T | 21.51 | 58.11 | 1.86 | 8.86 | 6 | 50.21 | 99.71 | 7 |
| 20 | 4 | L | 10.82 | 27.44 | 2.34 | 8.47 | 5 | 56.93 | 89.74 | 10 |
| 20 | 6 | T | 35.60 | 85.72 | 1.23 | 9.54 | 5 | 90.42 | 143.40 | 7 |
| 20 | 6 | L | 7.02 | 45.42 | 0.19 | 0.55 | 3 | 75.84 | 127.78 | 10 |
| 25 | 4 | T | 8.07 | 32.32 | 0.57 | 3.26 | 6 | 14.42 | 20.25 | 2 |
| 25 | 4 | L | 13.07 | 18.86 | 2.37 | 3.96 | 0 | 52.07 | 71.69 | 9 |
| 25 | 6 | T | 39.25 | 68.34 | 7.64 | 30.26 | 1 | 49.33 | 61.64 | 2 |
| 25 | 6 | L | 37.46 | 39.49 | 11.14 | 28.24 | 1 | 108.43 | 128.53 | 9 |
| 30 | 4 | T | 15.72 | 51.52 | 2.10 | 7.06 | 5 | infeas | infeas | 0 |
| 30 | 4 | L | 33.40 | 41.75 | 4.74 | 8.66 | 2 | 48.69 | 73.76 | 8 |
| 30 | 6 | T | 23.53 | 68.66 | 1.50 | 5.27 | 1 | infeas | infeas | 0 |
| 30 | 6 | L | - | - | - | - | 0 | - | - | 8 |
| 40 | 4 | T | 14.62 | 38.12 | 2.45 | 6.42 | 0 | 36.03 | 44.48 | 3 |
| 40 | 4 | L | - | - | - | - | 1 | - | - | 8 |
| 40 | 6 | T | 21.61 | 44.08 | 6.10 | 25.65 | 1 | 74.28 | 86.86 | 3 |
| 40 | 6 | L | - | - | - | - | 0 | - | - | 7 |
| 50 | 4 | T | 15.81 | 15.81 | 0.90 | 0.90 | 1 | infeas | infeas | 0 |
| 50 | 4 | L | - | - | - | - | 0 | - | - | 0 |
| 50 | 6 | T | 19.50 | 27.85 | 3.25 | 8.31 | 2 | infeas | infeas | 0 |
| 50 | 6 | L | - | - | - | - | 0 | - | - | 0 |
| 75 | 10 | T | - | - | - | - | 0 | - | - | 0 |
| 75 | 10 | L | - | - | - | - | 0 | - | - | 0 |
| 100 | 10 | T | - | - | - | - | 0 | - | - | 0 |
| 100 | 10 | L | - | - | - | - | 0 | - | - | 0 |

For instances where the exact SP model could be solved to near-optimality (optimality gap < 2%), the "Best of 4" matheuristic demonstrates very high accuracy. Its average deviations from the best-known SP solution are minimal, frequently falling below 5%. In contrast, the Rule-based heuristic performs poorly, with average deviations often exceeding 50% and reaching maximum of 108%, $n = 15$ in tight and 129%, $n = 25$ in loose configurations. These results confirm that while the Rule-based heuristic can provide an initial feasible configuration, it is insufficient for achieving competitive cycle time minimization.

For the remaining instances where the SP model cannot be solved to near-optimality (optimality gap > 2%), performance is evaluated through relative comparisons between heuristics. The following measures are employed:

- $(\text{Rule-based heuristic} - \text{"Best of 4"}) / \text{"Best of 4"}:$

This metric quantifies the performance gap between the simple heuristic, Rule-based and the most sophisticated matheuristic, "Best of 4". High values here indicate that the simple rules are insufficient.

- $(\text{Rule-based heuristic} - \text{"Best of 2"}) / \text{"Best of 2"}:$

This metric calculates how much the Rule-based heuristic lags behind the intermediate "Best of 2" matheuristic.

- $(\text{"Best of 2"} - \text{"Best of 4"}) / \text{"Best of 4"}:$

This metric isolates the "value add" of the additional improvement procedures included in the "Best of 4" matheuristic. It answers whether the extra computational time yields better solutions.

Comparative results for these instances are reported in Table 64.

Table 64: Relative percent deviations when optimality gap > 2%

| <i>n</i> | <i>m</i> | Setup | (Rule-based – “Best of 4”) / “Best of 4” | | (Rule-based – “Best of 2”) / “Best of 2” | | Freq. Feas |
|----------|----------|-------|--|--------|--|--------|------------|
| | | | Avg | Max | Avg | Max | |
| 15 | 4 | T | - | - | - | - | 9 |
| 15 | 4 | L | - | - | - | - | 10 |
| 15 | 6 | T | - | - | - | - | 9 |
| 15 | 6 | L | - | - | - | - | 10 |
| 20 | 4 | T | - | - | - | - | 7 |
| 20 | 4 | L | 14.39 | 14.39 | 8.40 | 8.40 | 10 |
| 20 | 6 | T | - | - | - | - | 7 |
| 20 | 6 | L | - | - | - | - | 10 |
| 25 | 4 | T | - | - | - | - | 2 |
| 25 | 4 | L | 32.02 | 36.50 | 14.56 | 27.65 | 9 |
| 25 | 6 | T | - | - | - | - | 2 |
| 25 | 6 | L | 69.27 | 97.75 | 43.03 | 64.60 | 9 |
| 30 | 4 | T | - | - | - | - | 0 |
| 30 | 4 | L | 39.12 | 57.37 | 26.29 | 47.06 | 8 |
| 30 | 6 | T | infeas | infeas | infeas | infeas | 0 |
| 30 | 6 | L | 62.42 | 88.33 | 33.28 | 59.77 | 8 |
| 40 | 4 | T | 29.71 | 29.71 | 29.71 | 29.71 | 3 |
| 40 | 4 | L | 32.90 | 41.61 | 18.91 | 30.53 | 8 |
| 40 | 6 | T | infeas | infeas | infeas | infeas | 3 |
| 40 | 6 | L | 67.24 | 76.21 | 31.45 | 68.30 | 7 |
| 50 | 4 | T | infeas | infeas | infeas | infeas | 0 |
| 50 | 4 | L | infeas | infeas | infeas | infeas | 0 |
| 50 | 6 | T | infeas | infeas | infeas | infeas | 0 |
| 50 | 6 | L | infeas | infeas | infeas | infeas | 0 |
| 75 | 10 | T | 75.19 | 91.32 | 46.49 | 73.02 | 6 |
| 75 | 10 | L | 79.66 | 96.42 | 56.16 | 91.67 | 6 |
| 100 | 10 | T | 45.20 | 57.12 | 20.74 | 39.18 | 5 |
| 100 | 10 | L | 57.04 | 66.23 | 38.48 | 61.51 | 4 |

Results in Table 64 indicates that, for instances where the exact SP model could not be solved to near-optimality (optimality gap > 2%), the “Best of 4” matheuristic serves as the performance benchmark. In contrast, the Rule-based heuristic performs poorly relative to this baseline, with average deviations frequently exceeding 30% and reaching nearly 80% in loose configurations.

For large-sized instances ($n = 75, 100$) where exact solutions are unavailable, the Rule-based heuristic lags significantly, showing average performance gaps of 45% to 80% compared to the “Best of 4” baseline. Notably, as the problem size increases to $n = 75$, the gap between the Rule-based heuristic and the “Best of 2” strategy is also substantial, reaching approximately 46% for tight setups. This

suggests that for the largest, most complex industrial cases, the comprehensive search performed by the “Best of 4” strategy is strictly necessary to maintain solution quality.

To isolate the contribution of additional improvement procedures, Table 65 reports the relative performance of the “Best of 2” strategy with respect to the “Best of 4” strategy.

Table 65: Relative percent deviations of matheuristics when optimality gap > 2%

| <i>n</i> | <i>m</i> | Setup | ("Best of 2" – "Best of 4")/ "Best of 4" | |
|----------|----------|-------|--|-------|
| | | | Avg | Max |
| 15 | 4 | T | - | - |
| 15 | 4 | L | - | - |
| 15 | 6 | T | - | - |
| 15 | 6 | L | - | - |
| 20 | 4 | T | - | - |
| 20 | 4 | L | 5.52 | 5.52 |
| 20 | 6 | T | - | - |
| 20 | 6 | L | - | - |
| 25 | 4 | T | - | - |
| 25 | 4 | L | 12.35 | 30.82 |
| 25 | 6 | T | - | - |
| 25 | 6 | L | 18.32 | 36.55 |
| 30 | 4 | T | - | - |
| 30 | 4 | L | 9.30 | 30.44 |
| 30 | 6 | T | 5.72 | 5.72 |
| 30 | 6 | L | 26.29 | 73.75 |
| 40 | 4 | T | 0.00 | 0.0 |
| 40 | 4 | L | 11.73 | 21.59 |
| 40 | 6 | T | 7.54 | 8.22 |
| 40 | 6 | L | 27.30 | 62.88 |
| 50 | 4 | T | 9.91 | 20.89 |
| 50 | 4 | L | 7.43 | 15.13 |
| 50 | 6 | T | 27.64 | 41.27 |
| 50 | 6 | L | 19.10 | 36.63 |
| 75 | 10 | T | 21.08 | 39.14 |
| 75 | 10 | L | 20.79 | 53.04 |
| 100 | 10 | T | 19.31 | 48.48 |
| 100 | 10 | L | 16.07 | 35.70 |

Results in Table 65 demonstrate the relative performance loss of the “Best of 2” matheuristic when compared to “Best of 4” matheuristic. For small-sized instances the “Best of 2” matheuristic is sufficient enough. In contrast, as problem complexity increases, the “Best of 2” matheuristic begins to lag significantly. This

trend is particularly acute in loose configurations for medium-sized instances ($n = 30, 40$), where average deviations frequently exceed 25% and maximum deviations become 73.75% ($n = 30, m = 6$). For large-sized instances $n = 75, 100$, the “Best of 2” matheuristic consistently underperforms. In these cases, it shows average performance gaps ranging from 16% to 21% and maximum deviations reaching approximately 48% to 53% compared to the “Best of 4” matheuristic. This data suggests that while “Best of 2” matheuristic is viable for smaller, tighter problems, the deeper search capabilities of “Best of 4” matheuristic are essential for maintaining high solution quality in large industrial systems.

CHAPTER 7

CONCLUSION

This thesis addressed the Type-II Transfer Line Balancing Problem by explicitly incorporating uncertainty associated with stochastic workstation maintenance times. Maintenance activities introduce variability that deterministic models cannot adequately capture; therefore, a scenario-based stochastic programming formulation and a constraint programming model were developed to minimize expected cycle time subject to technological, precedence, and workstation exclusion and inclusion constraints. Computational results show that while exact methods are effective for small-to-medium-sized instances, problem complexity increases rapidly with instance size, confirming the need for scalable solution approaches.

To address these computational challenges, a specialized Branch and Bound algorithm with problem-specific lower bounds and dominance rules was developed. Furthermore, heuristic extensions were introduced, including an alpha-percent Branch and Bound heuristic and model-based matheuristic procedures. The matheuristic framework applies decomposition-based block formation and workstation assignment, followed by local improvement mechanisms such as workstation reallocation and block or operation exchanges. These approaches provide high-quality solutions for large-scale instances where exact methods become computationally impractical.

The experimental analysis demonstrates that explicitly modeling maintenance uncertainty significantly affects both system configuration and performance outcomes, and that deterministic approximations may underestimate expected cycle times. By integrating stochastic modeling with exact and matheuristic optimization strategies, this study provides a comprehensive methodological framework for

balancing transfer lines under uncertainty and narrows the gap between theoretical models and practical manufacturing conditions.

Future research may extend this framework to reconfigurable manufacturing systems by incorporating structural adjustment decisions and their associated costs into stochastic transfer line design, enabling configurations that remain efficient under changing operating conditions.

Another promising direction for future research is the inclusion of secondary criteria, such as minimizing total cost, and the study of transfer line balancing problems within a multi-criteria optimization context.

REFERENCES

- Andres, C., Miralles, C., & Pastor, R. (2008). Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research*, 187(3), 1212–1223. <https://doi.org/10.1016/j.ejor.2006.07.044>
- Arcus, A.L. (1966). COMSOAL: A computer method of sequencing operations for assembly lines. *International Journal of Production Research*, 4, 259–277. <https://doi.org/10.1080/00207546608943001>
- Battaia, O., Delorme, X., Dolgui, A., Frederic, G., & Finel, B. (2015). Flow line balancing problem: A survey. *2015 International Conference on Industrial Engineering and Systems Management (IESM)*. <https://doi.org/10.1109/IESM.2015.7380173>
- Battaia, O., & Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, 142(2), 259–277. <https://doi.org/10.1016/j.ijpe.2012.10.020>
- Battaia, O., Dolgui, A., & Guschinsky, N. (2016). Integrated process planning and system configuration for mixed-model machining on rotary transfer machine. *International Journal of Computer Integrated Manufacturing*, 30, 910–925. <https://doi.org/10.1080/0951192X.2016.1193219>
- Battaia, O., Dolgui, A., & Guschinsky, N. (2023). MIP-based heuristics for combinatorial design of reconfigurable rotary transfer machines for production of multiple parts. *International Journal of Production Economics*, 262, 108904. <https://doi.org/10.1016/j.ijpe.2023.108904>
- Beldar, P., Fathi, M., Nourmohammadi, A., Delorme, X., Battaia, O., & Dolgui, A. (2025). Transfer line balancing problem: A comprehensive review, classification, and Research Avenues. *Computers & Industrial Engineering*, 201, 110913. <https://doi.org/10.1016/j.cie.2025.110913>
- Belmokhtar, S., Bratcu, A. I., & Dolgui, A. (2006). Modular Machining Line Design and Reconfiguration: Some Optimization Methods. *Manufacturing the Future*. <https://doi.org/10.5772/5047>
- Borisovsky, P. A., Delorme, X., & Dolgui, A. (2013). Balancing reconfigurable machining lines via a set partitioning model. *International Journal of Production Research*, 52(13), 4026–4036. <https://doi.org/10.1080/00207543.2013.849857>
- Bratcu, A. I., Dolgui, A., & Belmokhtar, S. (2005). Reconfigurable transfer lines cost optimization - A linear programming approach. *IEEE Conference on*

Emerging Technologies and Factory Automation.
<https://doi.org/10.1109/etfa.2005.1612581>

- Cakir, B., Altiparmak, F., & Dengiz, B. (2011). Multi-objective optimization of a stochastic assembly line balancing: A hybrid simulated annealing algorithm. *Computers & Industrial Engineering*, 60(3), 376–384. <https://doi.org/10.1016/j.cie.2010.08.013>
- Delorme, X., Malyutin, S., & Dolgui, A. (2016). A multi-objective approach for design of reconfigurable transfer lines. *IFAC-PapersOnLine*, 49(12), 509–514. <https://doi.org/10.1016/j.ifacol.2016.07.675>
- Dinler, D., & Tural, M. K. (2021). Exact solution approaches for the workload smoothing in Assembly Lines. *Engineering Science and Technology, an International Journal*, 24(6), 1318–1328. <https://doi.org/10.1016/j.jestch.2021.03.013>
- Dolgui, A. (2009). An approach to transfer line balancing via a special set partitioning problem. *13th IFAC Symposium on Information Control Problems in Manufacturing*. <https://doi.org/10.3182/20090603-3-RU-2001.0064>
- Dolgui, A. (2010). Post-optimal analysis for a design problem of machining lines. *IFAC Proceedings Volumes*, 43(17), 256–260. <https://doi.org/10.3182/20100908-3-IT-2016.00044>
- Dolgui, A., Finel, B., Guschinsky, N., Levin, G., & Vernadat, F. (2005a). A heuristic approach for transfer line balancing. *Journal of Intelligent Manufacturing*, 16, 159–171. <https://doi.org/10.1007/s10845-005-5519-9>
- Dolgui, A., Guschinsky, N., & Levin, G. (2005b). Transfer line balancing by a combined approach. *IFAC Proceedings Volumes*, 38, 277–282. <https://doi.org/10.3182/20050703-6-CZ-1902.00047>
- Dolgui, A., Finel, B., Guschinsky, N. N., Levin, G. M., & Vernadat, F. B. (2006). MIP approach to balancing transfer lines with blocks of parallel operations. *IIE Transactions*, 38(10), 869–882. <https://doi.org/10.1080/07408170500531334>
- Dolgui, A., Guschinsky, N., & Levin, G. M. (1999). On problem of optimal design of transfer lines with parallel and sequential operations. In *1999 7th IEEE International Conference on Emerging Technologies and Factory Automation Proceedings ETFA '99* (Vol. 1, pp. 329–334). <https://doi.org/10.1109/ETFA.1999.813402>
- Essafi, M., Delorme, X., Dolgui, A., & Guschinskaya, O. (2010). A MIP approach for balancing transfer line with complex industrial constraints. *Computers & Industrial Engineering*, 58, 393–400. <https://doi.org/10.1016/j.cie.2009.11.002>
- Essafi, M., Delorme, X., & Dolgui, A. (2012). A reactive grasp and path relinking for balancing reconfigurable transfer lines. *International Journal of*

- Production Research*, 50(18), 5213–5238.
<https://doi.org/10.1080/00207543.2012.677864>
- Guschinskaya, O., & Dolgui, A. (2006). A comparative evaluation of exact and heuristic methods for transfer line balancing problem. *IFAC Proceedings Volumes*, 39, 413–418. <https://doi.org/10.3182/20060517-3-FR-2903.00070>
- Guschinskaya, O., & Dolgui, A. (2009). Comparison of exact and heuristic methods for a transfer line balancing problem. *International Journal of Production Economics*, 120(2), 276–286.
<https://doi.org/10.1016/j.ijpe.2008.11.018>
- Guschinskaya, O., Dolgui, A., Guschinsky, N., & Levin, G. (2005). A hybrid approach for transfer line balancing. *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1, 311–316. <https://doi.org/10.1109/ETFA.2005.1612647>
- Hazır, Ö., & Dolgui, A. (2013). Assembly line balancing under uncertainty: Robust optimization models and exact solution method. *Computers & Industrial Engineering*, 65(2), 261–267. <https://doi.org/10.1016/j.cie.2013.03.004>
- Klein, R., & Scholl, A. (1996). Maximizing the production rate in simple assembly line balancing – A branch and bound procedure. *European Journal of Operational Research*, 91, 367–385. [https://doi.org/10.1016/0377-2217\(94\)00369-8](https://doi.org/10.1016/0377-2217(94)00369-8)
- Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., & Van Brussel, H. (1999). Reconfigurable Manufacturing Systems. *CIRP Annals*, 48(2), 527–540. [https://doi.org/10.1016/s0007-8506\(07\)63232-6](https://doi.org/10.1016/s0007-8506(07)63232-6)
- Krishnan, K. K., Almaktoom, A. T., & Udayakumar, P. (2016). Optimisation of stochastic assembly line for balancing under high variability. *International Journal of Industrial and Systems Engineering*, 22(4), 440. <https://doi.org/10.1504/ijise.2016.075205>
- Lahrichi, Y., Grangeon, N., Deroussi, L., & Norre, S. (2020). A new split-based hybrid metaheuristic for the reconfigurable transfer line balancing problem. *International Journal of Production Research*, 59(4), 1127–1144. <https://doi.org/10.1080/00207543.2020.1720929>
- Li, Y., Saldanha-da-Gama, F., Liu, M., & Yang, Z. (2022). A risk-averse two-stage stochastic programming model for a joint multi-item capacitated line balancing and lot-sizing problem. *European Journal of Operational Research*, 304(1), 353–365. <https://doi.org/10.1016/j.ejor.2021.09.043>
- Liu, S. B., Ong, H. L., & Huang, H. C. (2004). A bidirectional heuristic for stochastic assembly line balancing type II problem. *The International Journal of Advanced Manufacturing Technology*, 25(1-2), 71–77. <https://doi.org/10.1007/s00170-003-1833-5>

- Liu, X., Chen, J., & Li, A. (2019). Optimisation of line configuration and balancing for reconfigurable transfer lines considering demand uncertainty. *International Journal of Production Research*, 59(2), 444–466. <https://doi.org/10.1080/00207543.2019.1696490>
- Lolli, F., Balugani, E., Gamberini, R., & Rimini, B. (2017). Stochastic assembly line balancing with learning effects. *IFAC-PapersOnLine*, 50(1), 5706–5711. <https://doi.org/10.1016/j.ifacol.2017.08.1122>
- Lyu, J. (1997). A single-run optimization algorithm for stochastic assembly line balancing problems. *Journal of Manufacturing Systems*, 16(3), 204–210. [https://doi.org/10.1016/s0278-6125\(97\)88888-7](https://doi.org/10.1016/s0278-6125(97)88888-7)
- Mehrabi, M. G., Ulsoy, A. G., & Koren, Y. (2000). Reconfigurable manufacturing systems: Key to future manufacturing. *Journal of Intelligent Manufacturing*, 11(4), 403–419. <https://doi.org/10.1023/a:1008930403506>
- Patterson, J. H., & Albracht, J. J. (1975). Technical Note: Assembly-line balancing: zero-one programming with fibonacci search. *Operations Research*, 23, 166–172. <https://doi.org/10.1287/opre.23.1.166>
- Roy, D. (2011). Optimum assembly line balancing: A stochastic programming approach. *International Journal of Industrial Engineering Computations*, 2(2), 329–336. <https://doi.org/10.5267/j.ijiec.2010.04.001>
- Salveson, M. E. (1955). The assembly-line balancing problem. *Journal of Fluids Engineering*, 77(6), 939–947. <https://doi.org/10.1115/1.4014559>
- Sarin, S. C., Erel, E., & Dar-El, E. M. (1999). A methodology for solving single-model, stochastic assembly line balancing problem. *Omega*, 27(5), 525–535. [https://doi.org/10.1016/s0305-0483\(99\)00016-x](https://doi.org/10.1016/s0305-0483(99)00016-x)
- Shin, D., & Min, H. (1991). Uniform Assembly Line Balancing with stochastic task times in just-in-time manufacturing. *International Journal of Operations and Production Management*, 11(8), 23–34. <https://doi.org/10.1108/eum0000000001278>
- Telemeci, Y. E., & Azizoğlu, M. (2022). An exact solution approach to the type-II transfer line balancing problem. *IFAC-PapersOnLine*, 55(10), 442–447. <https://doi.org/10.1016/j.ifacol.2022.09.433>
- Telemeci, Y.E., & Azizoğlu, M. (2024). Type-II transfer line balancing problem – a branch and bound approach. *Computers & Industrial Engineering*, 198, 110689. <https://doi.org/10.1016/j.cie.2024.110689>

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Telemeci, Yasin Ersin

EDUCATION

| Degree | Institution | Year of Graduation |
|-------------|-------------------------------|--------------------|
| MS | METU Industrial Engineering | 2019 |
| BS | METU Industrial Engineering | 2016 |
| High School | Çanakkale Science High School | 2011 |

WORK EXPERIENCE

| Year | Place | Enrollment |
|--------------|--------------|-----------------------------|
| 2016-Present | ASELSAN A.Ş. | Production Planning Manager |

FOREIGN LANGUAGES

Advanced English, Basic German

PUBLICATIONS

1. Telemeci, Y.E., & Azizoğlu, M. (2022). An Exact Solution Approach to the Type-II Transfer Line Balancing Problem. IFAC-PapersOnLine.
2. Telemeci, Y.E., & Azizoğlu, M. (2024). Type-II transfer line Balancing problem - A branch and bound approach. Comput. Ind. Eng., 198, 110689.

HOBBIES

Tennis, Ski, Football, Reading, Traveling