الجمهورية الجزائرية الديمقراطية الشعبية

People's Democratic Republic of Algeria

وزارة التعليم العالي والبحث العلمي

Ministry of Higher Education and Scientific Research

University of
Saad Dahleb
Blida 1

جـــامـــــــعـــة
ســـــعد دحـــــلب
البـــــليـــــدة 1

Faculty of Science

Department of Mathematics

A thesis submitted in partial fulfillment of the requirements for the Master's degree

Major: Operational Research

Topic:

Optimizing Cut Order Planning in the Garment Industry using Mixed-Integer Nonlinear Programming with Open-Source Solvers

Ayadi Maria

In front of the jury composed of:

Mr. R. Frihi	MCB	Univ-blida 1	President
Mrs. S. Arrache	MAA	Univ-blida 1	Examiner
Mr. R. Boudjemaa	PR	Univ-blida 1	Supervisor

2024/2025

Acknowledgments

﴿وَقُل رَّبِّ زِدْنِي عِلْمًا﴾
سورة طه، الآية *114*

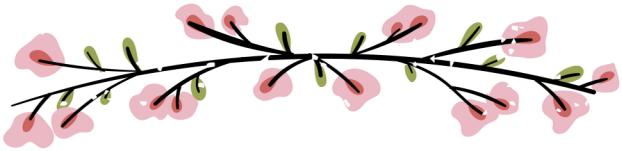
"And say, 'My Lord, increase me in knowledge.'"
Surah Taha, Ayah 114

First and foremost, I thank **Allah** for granting me the strength, patience, and perseverance to complete this thesis.

I would like to express my sincere gratitude to my supervisor, **Mr. Redouane Boudjemaa**, for his invaluable guidance, encouragement, and continuous support throughout this work. His insightful feedback and constant availability were essential to the progress of this research.

My heartfelt thanks also go to the honorable jury members, **Mr.Redouane Frihi** and **Mrs. SAIDA ARRACHE**, for accepting to evaluate this thesis and for their constructive remarks and suggestions.

Finally, I am deeply grateful to my **parents**, **relatives**, and **friends** for their endless love, patience, and encouragement throughout my academic journey**.**



Dedication

To my dearest **parents**, whose unwavering love, endless encouragement, and countless sacrifices laid the foundation for everything I am today. Your belief in me has been my constant motivation.

To my wonderful siblings, **Lina**, **Ayoub**, and **Bissane**, thank you for the laughter, the support, and for always being my first friends. Your presence in my life is a true blessing.

And to my incredible friends, **Nyhel**, **Radja**, **Soumia**, and **Riham**, thank you for the camaraderie, the understanding, and for making this journey so much brighter. Your friendship has been an invaluable source of strength and joy. This achievement is as much yours as it is mine.

Last but not least I want to thank me, I want to thank me for believing in me, I want to thank me for doing all this hard work.

Maria Ayadi

Abstract

The aim of this thesis is to develop and implement a mathematical model to optimize the fabric usage in garment manufacturing through Cut Order Planning (COP). The proposed model is based on a Mixed-Integer Nonlinear Programming (MINLP) formulation that integrates discrete decisions such as pile count and size assignment with nonlinear constraints related to fabric consumption. This model is inspired by the work of Ünal and Yüksel (2020) and reimplemented in an open-source environment using Pyomo and the SCIP solver.

The approach allows for minimizing fabric waste while satisfying production constraints across multiple product types. To evaluate the efficiency and practicality of the solution, results were obtained for shirts, trousers, sweatshirts, and coats. A comparison with the original LINGO-based model was conducted, focusing on iteration count and constraint satisfaction, while taking hardware differences into account.

Keywords:

Cut Order Planning, Mixed Integer Nonlinear Programming, Fabric Optimization, Garment Industry, SCIP, Open-Source Optimization.

Résumé

L'objectif de ce mémoire est de développer et de mettre en œuvre un modèle mathématique visant à optimiser l'utilisation du tissu dans l'industrie de l'habillement, à travers le processus de planification des ordres de coupe (Cut Order Planning – COP). Le modèle proposé repose sur une formulation en Programmation Non Linéaire en Nombres Mixtes (MINLP), intégrant à la fois des décisions discrètes (comme le nombre de couches et l'affectation des tailles) et des contraintes non linéaires liées à la consommation de tissu. Ce modèle s'inspire des travaux de Ünal et Yüksel (2020) et a été réimplémenté dans un environnement open-source en utilisant Pyomo et le solveur SCIP.

L 'approche permet de minimiser le gaspillage de tissu tout en respectant les contraintes de production sur plusieurs types de vêtements. Les performances du modèle ont été évaluées à travers des cas tests (chemises, pantalons, sweatshirts, manteaux). Une comparaison avec le modèle original basé sur LINGO a été réalisée, en mettant l'accent sur le nombre d'itérations et la satisfaction des contraintes, tout en tenant compte des différences matérielles.

Mots-clés :

Planification des ordres de coupe, Programmation Non Linéaire en Nombres Mixtes, Optimisation du tissu, Industrie de l'habillement, SCIP, Optimisation open-source.

الملخص

يهدف هذا البحث إلى تطوير وتنفيذ نموذج رياضي يهدف إلى تحسين الاستخدام الأمثل للقماش في صناعة الملابس، وذلك من خلال عملية تخطيط أوامر القص(Cut Order Planning – COP). يعتمد النموذج المقترح على صياغة ضمن البرمجة غير الخطية بالأعداد المختلطة(MINLP) ، والتي تدمج كلاً من القرارات المنفصلة (مثل عدد الطبقات وتخصيص الأحجام) والقيود غير الخطية المتعلقة باستهلاك القماش. يستلهم هذا النموذج أعمال Ünal و Yüksel (2020)وقد أعيد تنفيذه في بيئة مفتوحة المصدر باستخدام Pyomoوحل المشاكل SCIP (SCIP Solver).

يتيح هذا النهج تقليل هدر القماش إلى أدنى حد مع احترام قيود الإنتاج على أنواع متعددة من الملابس. تم تقييم أداء النموذج من خلال حالات اختبار (مثل القمصان، البنطلونات، السترات الثقيلة، والمعاطف). كما تم إجراء مقارنة مع النموذج الأصلي المعتمد علىLINGO ، مع التركيز على عدد التكرارات واستيفاء القيود، مع الأخذ في الاعتبار الاختلافات المادية.

الكلمات المفتاحية:

تخطيط أوامر القص، البرمجة غير الخطية بالأعداد المختلطة، تحسين استخدام القماش، صناعة الملابس، SCIP، التحسين مفتوح المصدر.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ABBREVIATIONS AND NOTATIONS

Abbreviations

COP	Cut Order Planning.
LP	Linear Programming.
NLP	Non-Linear Programming.
ILP	Integer Linear Programming.
MILP	Mixed Integer Linear Programming.
MINLP	Mixed Integer Non Linear Programming.
NP-hard	Non-deterministic Polynomial-time hard.
NP-complete	Non-deterministic Polynomial-time complete.
SCIP	Solving Constraint Integer Programs.
CAD	Computer-Aided Design.
ERP	Enterprise Resource Planning.
SKU	Stock Keeping Units.

Notations

$\mathbb{R}$	Set of real numbers

GENERAL INTRODUCTION

Operations Research (OR) uses quantitative methods and analytical tools to help decision-makers optimize the performance of various systems, including those in financial, scientific, and industrial sectors. The goal of OR is to enhance system efficiencies through a systematic and scientific approach. The formal beginnings of OR can be traced back to World War II, when it was employed to improve military operations. The British military worked alongside scientists to optimize resource allocation and manage operations effectively, notably utilizing radar technology to monitor aircraft. This success paved the way for the application of OR in business, industry, and government in the post-war era. The expansion of OR was further accelerated by the rise of high-speed computers, which made it possible to perform the complex calculations necessary for OR techniques. Professional organizations such as the Operational Research Society in Britain and the Operations Research Society of America (ORSA) were founded, eventually merging to form INFORMS. Today, OR plays a crucial role in various fields, boosting efficiency and productivity. It continues to adapt and grow, driven by advancements in computational methods and the increasing complexity of organizational challenges. By offering quantitative insights and supporting informed decision-making, OR remains an essential discipline in contemporary society, improving quality of life and organizational effectiveness.

Within this context, the apparel manufacturing industry faces growing pressure to optimize production while managing diverse customer demands, tight deadlines, and rising material costs. One of the most fabric- and cost-intensive stages in apparel production is the cutting department, where raw fabrics are spread and cut into specific garment components. The planning process for this operation is known as Cut Order Planning (COP), and it plays a critical role in reducing fabric waste and improving

productivity.

The COP problem is combinatorial in nature and subject to numerous constraints such as fabric lay length, number of plies, and exact demand fulfillment for different garment sizes. Due to these complexities, traditional manual methods or spreadsheet-based solutions are often inefficient. Recent research has proposed mathematical optimization as a powerful alternative.

This thesis builds upon the study conducted by Ünal and Yüksel, who formulated the Cut Order Planning problem as a Mixed-Integer Nonlinear Programming (MINLP) model and solved it using the LINGO solver. Their work demonstrated significant fabric savings, but it relied on proprietary software, limiting reproducibility and accessibility for small-to-medium enterprises. To overcome this limitation, our study re-implements the model in Python using Pyomo and evaluates the performance of open-source solvers such as SCIP, Bonmin, and Couenne.

The thesis is structured as follows:

- Chapter 1 presents a detailed literature review on Cut Order Planning (COP). It explores the operational context of COP in the apparel industry, outlines key challenges, and identifies research gaps. Particular attention is given to solver limitations and the lack of accessible, open-source implementations.

- Chapter 2 outlines the evolution of optimization techniques, starting from Linear Programming (LP) and advancing through Nonlinear Programming (NLP), Integer Programming (ILP), and Mixed-Integer Linear Programming (MILP), before reaching the complexity of MINLP. Each model is discussed in terms of mathematical formulation, solution strategies, and computational complexity, establishing the theoretical foundation required to address the COP problem.

- Chapter 3 defines the mathematical model used in this thesis. It re-implements the MINLP formulation originally proposed by Ünal and Yüksel, clearly presenting the objective function, decision variables, and constraints. The model is classified as NP-hard due to its combinatorial structure and nonlinear terms, and its implementation is validated using the open-source SCIP solver.

- Chapter 4 provides the core experimental work. It presents a real-world case study involving four garment types: shirts, coats, trousers, and sweatshirts. Results obtained using SCIP are compared to those from the original LINGO solver and further contrasted with Bonmin and Couenne. Solver efficiency, fabric savings, and runtime performance are analyzed and discussed in detail.

- The conclusion of the thesis summarizes the main findings, highlighting contri-

butions, and suggesting avenues for future research. It reflects on the practical value of transitioning from commercial solvers to open-source tools in apparel production environments.

In sum, this thesis contributes to the literature by offering a transparent, accessible, and efficient optimization-based approach to solving the COP problem. By leveraging open-source tools, it bridges the gap between academic research and real-world application, offering practical benefits to manufacturers seeking to optimize fabric usage and improve sustainability.

CHAPTER 1

LITERATURE REVIEW ON CUT ORDER PLANNING

1.1 Introduction

This chapter provides a structured review of the literature related to Cut Order Planning (COP) in the apparel industry. It begins with an overview of the apparel industry supply chain to contextualize where COP fits in the broader production process. Then, it presents a review of key academic works focused specifically on COP in the garment sector, highlighting various optimization approaches and practical applications.

The chapter continues with a detailed explanation of the core concepts and challenges involved in COP, followed by a discussion on how similar mathematical modeling techniques—such as MILP and MINLP—have been applied to non-garment cutting and packing problems. Finally, it summarizes the main research gaps identified in the literature, which this thesis aims to address through an open-source and MINLP-based approach.

1.2 Overview of the Apparel Industry Supply Chain

The apparel industry is a global and rapidly evolving sector that has transitioned from local, manual production to complex, internationally distributed supply chains. According to [6], this shift was driven by technological advances and the pursuit of lower production costs through offshoring. Today's apparel supply chains must balance cost-efficiency with agility to respond to fast-changing consumer trends.

One of the most critical stages in the apparel manufacturing process is the cutting phase, where fabric is prepared based on demand for various styles, sizes, and colors.

This stage plays a major role in overall production cost and efficiency. Poor planning at this point can result in substantial fabric waste, increased labor hours, and production delays.

One of the most planning-intensive operations in this stage is Cut Order Planning (COP), a process that will be explored in detail in the following section.

COP involves key steps such as marker making, fabric spreading, and cutting. A marker is a layout plan that minimizes fabric waste by determining the most efficient arrangement of garment pieces. Each marker must consider variations in size, style, and color, adding to the complexity of planning.

To meet the demands of modern supply chains, many companies are adopting digital tools such as CAD(Computer-Aided Design) for marker planning and ERP(Enterprise Resource Planning) systems for production coordination. Optimization-based decision support systems are also gaining popularity to improve the precision and efficiency of COP.

In summary, the apparel industry's move toward faster and more flexible production systems has elevated the importance of planning operations like COP. As production complexity grows, so too does the need for optimization techniques that can support smarter, data-driven decisions.

1.3 Literature Review on Cut Order Planning (COP) in the Garment Industry

Cut Order Planning (COP) has emerged as one of the most critical planning activities in garment manufacturing. Its main purpose is to determine how customer orders—often diverse in size, style, and color—can be translated into efficient cutting operations that minimize fabric waste while respecting production constraints. Given the fabric cost's significant share of total production expenses, COP optimization has attracted considerable academic attention.

Rose and Shier [45] were among the first to apply optimization techniques to COP by proposing a Mixed-Integer Linear Programming (MILP) model for cut scheduling in the apparel industry. Their work considered fabric width, style variety, and production constraints to minimize fabric waste, illustrating how MILP models could offer feasible solutions to real-world garment cutting problems. However, their use of commercial solvers limited the model's practical adoption in cost-sensitive environments.

Paşayev [42] conducted a field study in Turkish apparel factories, analyzing how production planning methods affected fabric cost. Although the study was non-mathematical, it showed that unoptimized COP operations could lead to over 20% excess fabric usage—highlighting the real-world consequences of inefficient planning and underscoring the need for more scientific approaches in this domain.

Utkün [49] emphasized the importance of marker planning on overall productivity, particularly in bathrobe production. The study demonstrated that differences in model design and marker layout strategies significantly impact fabric consumption. While this research didn't propose a new optimization model, it strongly reinforced the need for adaptive COP methods that respond to product variety and changing demand.

Kong et al. [33] introduced a MILP-based approach to optimize line balancing and production scheduling in garment factories. Although their model targeted sewing operations rather than cutting, the structural similarity of constraints—like production rates, resource limitations, and demand satisfaction—demonstrates the MILP framework's versatility in addressing apparel planning problems, including COP.

Liyanage et al. [38] took a different route by applying genetic programming to optimize workflows across multiple stages in textile manufacturing. Their research incorporated cutting, sewing, and finishing activities, showing that heuristic and metaheuristic methods like Genetic Algorithms can be used to reduce lead time and resource consumption—especially in environments with high complexity and uncertainty.

A more practice-oriented contribution is offered by LeanStitch platform [37], a website that explains the basics of COP implementation from an industrial perspective. The platform highlights practical concerns like lay planning, marker efficiency, and fabric utilization, aligning well with academic models while offering an accessible view for small and medium-sized factories.

Ünal and Yüksel [48] delivered one of the few studies using a full Mixed-Integer Nonlinear Programming (MINLP) approach. Their model captures the nonlinear relationships between marker length, fabric usage, and SKU (Stock Keeping Units) combinations, producing more accurate and realistic planning results. Their work demonstrated that MINLP provides superior modeling capability for COP compared to traditional MILP models—though it also requires more advanced solvers and computational power.

Finally, recent advances in artificial intelligence (AI) are beginning to reshape how decision-making is handled in the apparel supply chain. According to a white paper

by the Ghasemiran Foundation (2023) [21], AI techniques—including machine learning and reinforcement learning—can enhance demand forecasting, inventory management, and production scheduling in fashion industries. These techniques, when combined with mathematical models, hold strong potential to support intelligent COP systems that adapt to dynamic environments.

In summary, the body of research reviewed here spans from early MILP formulations to recent AI-driven approaches, illustrating a rich and evolving field. However, most studies still rely on proprietary tools, and few fully explore nonlinear models or open-source implementations. This thesis builds upon the work of Ünal and Yüksel by implementing their MINLP model in Python using the SCIP solver, aiming for better accessibility, reproducibility, and real-world relevance.

1.4 Cut Order Planning: Concepts and Challenges

Cut Order Planning (COP) is a core operation in apparel manufacturing that connects demand fulfillment with efficient fabric utilization. It involves determining how various customer orders—differentiated by style, size, and color—should be grouped and assigned to cutting markers to minimize fabric waste while satisfying production requirements.

The COP process starts after production orders are finalized and fabric rolls are available in the cutting department. The objective is to design marker plans and fabric lays that ensure all Stock Keeping Units (SKUs) are covered while minimizing the amount of fabric consumed. A marker is essentially a layout that determines how garment pieces are arranged on fabric spreads to maximize efficiency. COP thus answers key questions such as: how many garments of each type should be placed in each lay, how many markers are required, and how orders should be grouped to balance efficiency and feasibility.

The complexity of COP stems from several real-world operational constraints:

- **SKU Variability**: Orders include numerous combinations of sizes, colors, and styles, increasing the number of possible layout configurations.
- **Fabric Constraints**: Fabric rolls vary in width, shrinkage rate, or defect tolerance, all of which must be factored into planning.
- **Production Restrictions**: Machines and tables impose limits on lay length and layer height; certain garment types may require batching or specific equipment.

- **Exact Demand Fulfillment**: Overproduction is costly and underproduction is unacceptable; COP must meet demand precisely, sometimes within allowable tolerance margins.

According to Textile Engineering website [47], COP plays a decisive role in total manufacturing cost and speed. Efficient COP strategies reduce fabric waste, labor time, and machine downtime. Nevertheless, in many small and medium-sized apparel factories, COP is still carried out manually, relying on the planner's experience and heuristics. This often leads to suboptimal decisions, especially as product variety and order volume increase.

To better understand the practical implementation of COP, consider the schematic process flow shown in Figure 2.1. This diagram—adapted from Ünal and Yüksel [48]—illustrates the sequential steps in a typical cutting department, starting from fabric arrival to sorting and bundling of cut components.

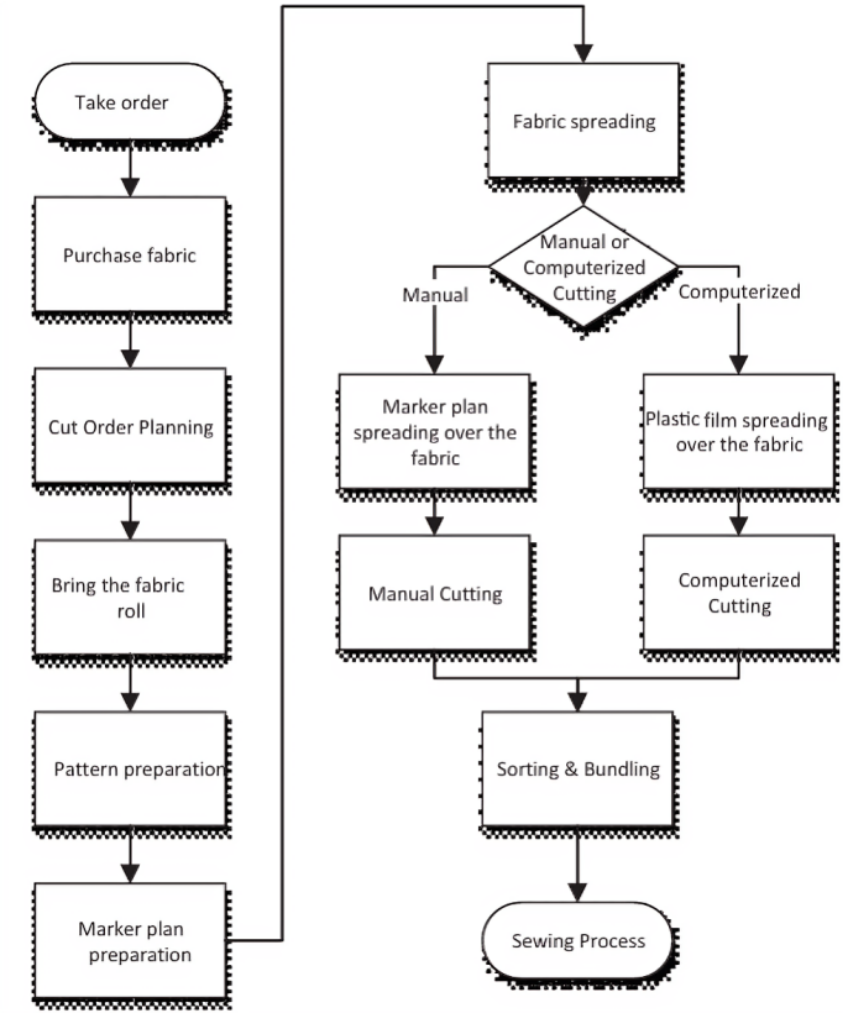


Figure 1.1: Typical process flow in a cutting department—from fabric arrival to sorting and bundling (adapted from Ünal & Yüksel [48]).

As depicted, the process begins with fabric roll reception and continues through quality checks, spreading, marker placement, cutting, sorting, and bundling. The figure highlights how tightly integrated COP is with other stages: poor COP decisions can create bottlenecks in later steps, leading to inefficiencies or material loss.

The COP process itself can be divided into four key planning phases:

1. **Order Grouping**: Determining which styles, sizes, and colors can be combined in the same lay to reduce the number of markers and maximize marker efficiency.

2. **Marker Planning**: Designing optimal markers to minimize fabric waste while accommodating garment shapes and sizes.

3. **Lay Planning**: Deciding how many fabric layers are needed for each marker, considering machine limits and fabric roll length.

4. **Execution**: Implementing the cutting operation according to the predefined plan, followed by sorting and bundling for further processing.

As emphasized by Ünal and Yüksel [48], each decision in the COP process directly affects key performance indicators such as fabric utilization, production throughput, and labor productivity. In high-volume manufacturing, the number of possible marker and lay combinations can reach into the thousands, making manual approaches increasingly inefficient and error-prone.

Consequently, the industry is seeing a shift toward algorithmic and data-driven approaches to COP. Mathematical optimization techniques—including Integer Linear Programming (ILP), Mixed-Integer Linear Programming (MILP), and Mixed-Integer Nonlinear Programming (MINLP)—are being explored to automate and enhance decision-making. These methods allow planners to handle the complex trade-offs involved in grouping, cutting, and allocation, all while adhering to operational constraints.

1.5 Applications of MILP and MINLP in Non-Garment Cutting Problems

While Cut Order Planning (COP) is specific to the apparel industry, its structure shares strong similarities with other cutting and packing problems commonly found in operations research. Several studies outside the textile sector have employed Mixed-Integer Linear Programming (MILP) and Mixed-Integer Nonlinear Programming (MINLP) to

model and solve such problems, demonstrating the robustness and versatility of these techniques.

One widely studied class of problems is the **cutting stock problem**, in which raw materials like metal, wood, or paper are cut into pieces of varying sizes to meet demand while minimizing waste. Wäscher et al. [51] provide a comprehensive typology of such problems, classifying them into one-dimensional, two-dimensional, and three-dimensional variants. Their work also distinguishes between cutting problems with fixed patterns and those that generate patterns dynamically, both of which are relevant to marker-making in garment COP.

In another domain, Berkey and Wang [11] developed MILP-based models for bin packing problems with fixed-plus-linear cost schemes. Their approach optimizes the use of bins with varying capacities while balancing fixed setup costs and variable loading costs. The model incorporates piece placement and bin utilization constraints similar in structure to those seen in COP. This study illustrates the ability of MILP to handle industrial settings involving discrete configurations and cost trade-offs—elements also central to cut planning in apparel.

Additionally, Grossmann [23] explored the broader field of MINLP techniques, discussing how they can address process systems engineering problems that involve nonlinear relationships and discrete decisions. His review highlights how MINLP offers significant modeling power for systems like blending, scheduling, and layout optimization—many of which present mathematical complexities analogous to those encountered in COP, such as bilinear terms and logical constraints.

A recent report by Avci and Topaloglu [5] presents a MINLP formulation for the trim-loss problem in the metal industry, where coils of sheet metal are cut to size with minimal waste. The authors emphasize that including nonlinearity in the model (e.g., in cost or cutting constraints) leads to more realistic and efficient outcomes. Their computational experiments also show that MINLP solvers like SCIP can deliver practical solutions within reasonable runtimes, reinforcing our thesis's use of SCIP for COP.

These examples collectively reinforce the relevance of MILP and MINLP for solving complex cutting and layout problems across industries. Since Cut Order Planning exhibits many of the same characteristics—integer-based decisions, nonlinear fabric consumption, and layout-dependent constraints—it is logical and methodologically sound to adopt similar optimization paradigms. This broader context supports the validity and transferability of our modeling approach to COP in the garment industry.

1.6 Summary of Gaps in Existing Research

While this chapter has reviewed a range of studies from both garment and non-garment industries—including MILP and MINLP applications in cutting and packing problems—certain methodological and practical gaps still persist in the academic literature on Cut Order Planning (COP):

- **Limited Solver Diversity**: Most published works rely on commercial solvers such as LINGO, Gurobi, or CPLEX. These tools, while powerful, often limit accessibility due to licensing costs. The underuse of open-source solvers like SCIP, which provide comparable performance and flexibility, represents a missed opportunity in COP research.
- **Underutilization of MINLP Frameworks**: Although COP naturally includes nonlinear constraints and integer decisions, the number of works employing full MINLP formulations remains limited. Many adopt MILP, which simplifies the model at the cost of realism. Ünal and Yüksel's use of MINLP was pioneering, but few have followed up on this approach, especially using open frameworks.
- **Computational Efficiency Overlooked**: Many studies focus on model structure or feasibility but neglect detailed performance benchmarks. In our work, solving the same MINLP model using SCIP instead of LINGO resulted in a noticeable reduction in the number of solver iterations and overall processing time. This highlights the value of solver selection and implementation strategy.
- **Reproducibility and Transparency**: Proprietary software restricts transparency and reproducibility. By re-implementing the model in Python with SCIP, our work enables other researchers and practitioners to replicate, validate, and extend our results without proprietary barriers.
- **Lack of Real-World Integration and Case-Based Evaluation**: Much of the existing literature relies on synthetic or simplified datasets. In contrast, this thesis directly applies the original data set from Ünal and Yüksel's study, allowing for direct comparison and real-world applicability.
- **Absence of Hybrid and Adaptive Methods**: Hybrid techniques that integrate heuristics with exact methods are rarely applied to COP. Similarly, adaptive systems that can respond to fluctuating demand or fabric availability remain underexplored.

- **Static Assumptions and Deterministic Models**: Most models assume fixed inputs and fail to account for real-time changes, demand uncertainty, or disruptions such as machine breakdowns or urgent order changes. The future of COP research lies in incorporating stochastic optimization, robust formulations, or even AI-based dynamic planning.

In light of these limitations, this thesis makes several contributions. It revisits a validated MINLP model, implements it using the open-source SCIP solver within a Python environment, and compares performance metrics with those obtained using LINGO. The improvements observed in computational efficiency, along with enhanced accessibility and reproducibility, demonstrate how such methodological refinements can benefit both research and practice.

1.7 Conclusion

This chapter presented a comprehensive overview of the Cut Order Planning (COP) process within the apparel supply chain, highlighting its significance as a critical driver of fabric efficiency and production performance. By tracing the evolution of the industry and outlining the operational challenges of COP—including SKU variability, fabric constraints, and manual dependency—the chapter established motivation for more advanced optimization techniques.

The literature review revealed that, although COP has been extensively studied, key research gaps remain. Most notably, prior work relies heavily on commercial solvers and simplified model structures (e.g., MILP rather than MINLP). Moreover, real-world data and dynamic modeling elements are often absent, limiting the practical relevance of many proposed approaches.

To address these limitations, this thesis builds upon the work of Ünal and Yüksel by re-implementing their MINLP model using an open-source optimization framework (SCIP). This improves accessibility, transparency, and enables more robust computational performance and scalability—particularly for large, complex garment planning scenarios. This review establishes a strong foundation for the methodological and computational contributions introduced in the following chapters. It supports the hypothesis that open-source solvers, when paired with well-formulated MINLP models, can provide industry-grade solutions to the COP problem without relying on proprietary tools.

CHAPTER 2

EVOLUTION OF OPTIMIZATION TECHNIQUES

2.1 Introduction

Optimization has long served as a foundational tool in operations research, enabling decision-makers to model and solve problems involving resource allocation, scheduling, and logistics. Over the decades, the field has evolved from early formulations of Linear Programming (LP) to more complex frameworks capable of handling discrete decisions and nonlinear system behaviors. This chapter presents a chronological and conceptual overview of this evolution—from Linear Programming (LP) to Mixed Integer Nonlinear Programming (MINLP).

As systems became more sophisticated, so did the mathematical tools required to represent them accurately. Real-world constraints are often nonlinear and involve binary or integer variables—features not addressable by LP or even traditional Nonlinear Programming (NLP). The development of Integer Linear Programming (ILP), Mixed-Integer Linear Programming (MILP), and eventually Mixed-Integer Nonlinear Programming (MINLP) addressed this need for expressive power.

Drawing on the seminal work of Grossmann [23], this chapter explores how these modeling frameworks have grown in capability, computational complexity, and practical relevance. We review their theoretical structures, discuss classical and modern solving techniques—including branch-and-bound, cutting planes, and hybrid heuristics—and compare them in terms of applicability, solver availability, and efficiency. This background lays the foundation for applying MINLP methods to the Cut Order Planning (COP) problem in the apparel industry in later chapters.

2.2 Optimization

Optimization is a cornerstone of operations research, representing a systematic approach to finding the most favorable solutions from a set of feasible alternatives. It is a mathematical and computational discipline that aims to either maximize or minimize an objective function while adhering to given constraints. With its ability to improve decision-making processes and allocate resources efficiently, optimization plays a critical role across diverse fields such as engineering, economics, logistics, and management.

Historically, the concept of optimization has its roots in the mathematical problem-solving techniques of ancient times. However, significant advancements were made in the 18th century when pioneers like Newton, Lagrange, and Cauchy developed methods using differential and variation calculus to solve optimization problems in physics and geometry. The field witnessed a revolutionary breakthrough in 1947, when George Dantzig [18] introduced the Simplex Method, which provided a systematic solution to linear programming problems.

Optimization is characterized by several essential components: an objective function that quantifies the goal to be achieved, decision variables that influence outcomes, and constraints that define the feasible solution space. Techniques such as linear programming, nonlinear programming, dynamic programming, and metaheuristic algorithms have been developed over the years to navigate the complexities of optimization problems and identify the best solutions [41].

Today, optimization is applied across various domains, including production planning, supply chain management, transportation, finance, energy, and healthcare. Its integration with cutting-edge technologies, such as artificial intelligence and machine learning, continues to expand its scope, enhancing decision-making and resource allocation in an ever-evolving world. By balancing competing objectives and constraints, optimization provides a robust framework to solve complex real-world problems and improve all efficiency.

2.3 Linear Programming

Linear programming, also known as linear optimization, involves maximizing or minimizing a linear objective function subject to a set of linear constraints. These constraints can be expressed as either equalities or inequalities. Typically, such optimiza-

tion problems are used to analyze scenarios involving profit and loss calculations.

Linear programming represents a crucial category of optimization problems. It helps identify the feasible region defined by the constraints and determines the optimal solution that yields the highest or lowest possible value of the objective function.

In simpler terms, linear programming is an optimization technique aimed at maximizing or minimizing a given objective function within a mathematical model. This model is governed by a set of requirements or restrictions, all expressed through linear relationships. The primary goal is to find the best possible solution that satisfies all these conditions.

2.3.1 Characteristics of Linear Programming

The following are the five characteristics of the linear programming problem:

Constraints – The limitations should be expressed in the mathematical form, regarding the resource.

Objective Function – In a problem, the objective function should be specified in a quantitative way.

Linearity – The relationship between two or more variables in the function must be linear. It means that the degree of the variable is one.

Finiteness – There should be finite and infinite input and output numbers. In case, if the function has infinite factors, the optimal solution is not feasible.

Non-negativity – The variable value should be positive or zero. It should not be a negative value.

Decision Variables – The decision variable will decide the output. It gives the ultimate solution of the problem. For any problem, the first step is to identify the decision variables.

2.3.2 General Form of LP

Linear Programming deals with optimizing a linear objective function subject to linear equality and inequality constraints. All decision variables are continuous (real numbers).

$$\text{Minimize or Maximize } Z = c^T x \tag{2.1}$$

Subject to:

$$Ax \leq b \tag{2.2}$$

$$Dx = e \tag{2.3}$$

$$x \geq 0 \tag{2.4}$$

where:

- n is the number of decision variables.
- m is the number of inequality constraints.
- p is the number of equality constraints.
- x is the vector of decision variables ($x \in \mathbb{R}^n$).
- c is the vector of coefficients for the objective function ($c \in \mathbb{R}^n$).
- A and D are matrices of coefficients for the inequality and equality constraints, respectively ($A \in \mathbb{R}^{m \times n}$ and $D \in \mathbb{R}^{p \times n}$).
- b and e are vectors of constants for the inequality and equality constraints, respectively ($b \in \mathbb{R}^m$ and $e \in \mathbb{R}^p$).
- $c^T x$ denotes the dot product (scalar product).

2.3.3 Limitations of LP in Real-World Problems

Linear programming (LP) offers a systematic approach to optimization; however, it comes with several limitations that restrict its application in real-world scenarios: such as its inability to handle integer solutions, non-linear relationships, and multi-objective scenarios. Due to that, we transition to mixed-integer linear programming (MILP) and mixed-integer non-linear programming (MINLP). These methods address real-world complexities more effectively, offering greater flexibility and practical applicability. This progression ensures optimization models align better with dynamic and diverse challenges.

2.3.4 Linear Programming Method

The linear programming problem can be solved using different methods, such as the graphical method, simplex method, or by using tools such as R, open solver etc. Here,

we will discuss the two most important techniques called the simplex method and graphical method in detail [16].

1. **Simplex Method Algorithm:**

   The simplex method is one of the most popular methods to solve linear programming problems. It is an iterative process to get the feasible optimal solution. In this method, the value of the basic variable keeps transforming to obtain the maximum value for the objective function. The algorithm for linear programming simplex method is provided below [16]:

   (a) **Step 1:** : Establish a given problem. (i.e) write the inequality constraints and objective function.

   (b) **Step 2:** Convert the given inequalities to equations by adding the slack variable to each inequality expression.

   (c) **Step 3:** Create the initial simplex table. Write the objective function at the bottom row. Here, each inequality constraint appears in its own row. Now, we can represent the problem in the form of an augmented matrix, which is called the initial simplex table.

   (d) **Step 4:** Identify the greatest negative entry in the bottom row, which helps to identify the pivot column. The greatest negative entry in the bottom row defines the largest coefficient in the objective function, which will help us to increase the value of the objective function as fastest as possible.

   (e) **Step 5:** Compute the quotients. To calculate the quotient, we need to divide the entries in the far right column by the entries in the first column, excluding the bottom row. The smallest quotient identifies the row. The row identified in this step and the element identified in the step will be taken as the pivot element.

   (f) **Step 6:** Carry out pivoting to make all other entries in column is zero.

   (g) **Step 7:** If there are no negative entries in the bottom row, end the process. Otherwise, start from step 4.

   (h) **Step 8:** Finally, determine the solution associated with the final simplex table.

2. **Graphical Method:**

   The graphical method is used to optimize the two-variable linear programming. If the problem has two decision variables, a graphical method is the best method to find the optimal solution. In this method, the set of inequalities are subjected to constraints. Then the inequalities are plotted in the XY plane. Once, all the inequalities are plotted in the XY graph, the intersecting region will help to decide the feasible region. The feasible region will provide the optimal solution as well as explains what all values our model can take. Detailed examples of the described methods are provided in [16].

2.4 Non-Linear Programming

Nonlinear Programming (NLP) is a field of mathematical optimization, specifically tackling problems where the objective function and/or constraints involve nonlinear behavior. It takes linear programming a step further by accommodating more intricate and realistic models that truly capture how real-world systems behave.

As optimization theory evolved in the mid-20th century, it became evident that many real-world challenges couldn't be effectively represented using just linear equations. In fields like engineering, economics, finance, and logistics, numerous systems display nonlinear characteristics—think economies of scale, nonlinear production rates, or curved cost functions.

While linear programming offered some powerful tools, it was confined to simpler scenarios. This limitation paved the way for nonlinear programming, which brought in a whole new level of modeling flexibility and accuracy.

2.4.1 General Form of NLP

Nonlinear Programming deals with optimizing an objective function subject to nonlinear and/or linear constraints. If either the objective function or any constraint is nonlinear, the problem is classified as a Nonlinear Programming (NLP) problem. All decision variables are continuous.

$$\text{Minimize or Maximize } f(x) \tag{2.5}$$

Subject to:

$$g_i(x) \leq 0 \quad \text{for } i = 1, \ldots, m_1 \tag{2.6}$$

$$h_j(x) = 0 \quad \text{for } j = 1, \ldots, m_2 \tag{2.7}$$

$$x \in X \subseteq \mathbb{R}^n \tag{2.8}$$

where:

- i is the index for inequality constraints ($i = 1, 2, \ldots, m_1$).
- j is the index for equality constraints ($j = 1, 2, \ldots, m_2$).
- x is the vector of decision variables ($x \in \mathbb{R}^n$).
- $f(x)$ the nonlinear objective function.
- $g_i(x)$ nonlinear inequality constraint functions.
- $h_j(x)$ nonlinear equality constraint functions.
- If $f(x)$, $g_i(x)$, and $h_j(x)$ are all convex (for minimization problems), the problem is a **Convex Nonlinear Program**, which is generally easier to solve. Otherwise, it's a **Non-convex Nonlinear**.

2.4.2 Nonlinear Programming Methods

Several methods have been developed to solve nonlinear programming problems, depending on the structure and complexity of the objective function and constraints. These include Gradient Descent, Newton-based methods, Interior Point methods, and metaheuristic approaches [2]. In practice, solvers such as `IPOPT`, `KNITRO`, `SCIP`, and `MATLAB`'s `fmincon` are commonly used for solving such problems efficiently [32].

- **Gradient-Based:**

  Gradient-based methods are iterative optimization techniques that use the gradient of the objective function to guide the search toward a minimum. At each step, the algorithm moves in the direction opposite to the gradient, aiming to reduce the function's value. The efficiency of these methods depends heavily on the choice of step size, which controls how far the algorithm moves at each iteration. These methods are well-suited for smooth, differentiable functions and are

commonly used in engineering and applied optimization. However, they may converge to local minima in non-convex problems. Despite this, they remain fundamental tools for solving large-scale nonlinear problems efficiently [7].

- **Interior-point methods:**

  Interior-point methods are optimization algorithms used in nonlinear programming to find solutions by iterating from within the feasible region. They incorporate barrier functions to manage inequality constraints and follow a central path to reach the optimal solution [35].

2.5 Integer Linear Programming

Integer Linear Programming (ILP) is an optimization technique that deals with problems involving linear relationships and integer variables. It extends Linear Programming (LP) by restricting decision variables to integer values. The objective in ILP is to find the optimal solution to a linear objective function while satisfying a set of linear constraints, with the added challenge of integer-only variables.

ILP finds applications in resource allocation, production planning, scheduling, network optimization, and other areas requiring discrete decisions.

2.5.1 General Form of ILP

Integer Linear Programming (ILP) is a subclass of linear programming where some or all of the decision variables are constrained to take integer values. When all decision variables are integers, the problem is called a pure ILP. If only a subset are integers and the rest are continuous, the model becomes a Mixed-Integer Linear Program (MILP).

The standard mathematical formulation of a pure ILP is:

$$\text{Minimize or Maximize } Z = c^T x \tag{2.9}$$

Subject to:

$$Ax \leq b \qquad \text{(Inequality constraints)} \tag{2.10}$$

$$Dx = e \qquad \text{(Equality constraints)} \tag{2.11}$$

$$x \in \mathbb{Z}^n \qquad \text{(Integer constraints)} \tag{2.12}$$

$$x \geq 0 \qquad \text{(Non-negativity)} \tag{2.13}$$

Where:

- $x \in \mathbb{Z}^n$: vector of integer decision variables.
- $c \in \mathbb{R}^n$: cost coefficients of the objective function.
- $A \in \mathbb{R}^{m_1 \times n}$, $D \in \mathbb{R}^{m_2 \times n}$: constraint coefficient matrices.
- $b \in \mathbb{R}^{m_1}$, $e \in \mathbb{R}^{m_2}$: right-hand side vectors.

Due to the discrete nature of decision variables, ILP problems are **NP-hard** [3], meaning that the solution time can grow exponentially with problem size. Consequently, specialized techniques like Branch and Bound, Cutting Planes, and Branch and Cut are used to solve them.

2.5.2 ILP Solution Algorithms

- **Branch-and-Bound:**

  A tree-based algorithm that solves Integer Linear Programs (ILPs) by recursively dividing the problem into subproblems (branching), solving their linear relaxations, and pruning branches that cannot yield better integer solutions (bounding) [36].

- **Cutting-Plane Method:**

  An iterative method that solves ILPs by solving the LP relaxation and progressively adding valid inequalities (cuts) to exclude fractional solutions, continuing until an integer optimal solution is found [36].

- **The Branch and Cut method:**

  Branch-and-Cut is a hybrid algorithm that combines branch-and-bound with cutting-plane techniques to solve Integer Linear Programming (ILP) and Mixed Integer Linear Programming (MILP) problems [31].

  It strengthens the linear programming relaxation by iteratively adding valid inequalities (cuts), while exploring the solution space via branching. This dual strategy makes Branch-and-Cut highly effective for large-scale combinatorial problems. The algorithm proceeds as follows: it solves LP relaxations of subproblems; if the solution is fractional, it attempts to generate cutting planes to eliminate it. If no cuts are found, the algorithm branches into subproblems. Integer feasible

solutions are compared and the best one is retained. This process repeats until all subproblems are either solved or pruned. Algorithm 1 outlines the full procedure:

Algorithm 1 Branch and Cut algorithm

```
1: Add the initial ILP to L, the list of active problems
2: Set x* = null and v* = −∞
3: while L is not empty do
4:     Select and remove (dequeue) a problem from L
5:     Solve the LP relaxation of the problem
6:     if the solution is infeasible then
7:         Go back to Step 3
8:     else
9:         Denote the solution by x with objective value v
10:        if v ≤ v* then
11:            Go back to Step 3
12:        end if
13:        if x is integer then
14:            Set v* ← v, x* ← x and go back to Step 3
15:        end if
16:        if desired, search for cutting planes that are violated by x then
17:            if any are found then
18:                Add them to the LP relaxation and return to Step 5
19:            end if
20:        end if
21:        Branch to partition the problem into new problems with restricted feasible regions. Add
       these problems to L and go back to Step 3
22:    end if
23: end while
24: return x*
```

Parameter Definitions:

L List (or queue) of active ILP subproblems to explore

x Current solution of the LP relaxation

x^* Best (incumbent) integer solution found so far

v Objective value of the current solution x

v^* Objective value of the incumbent solution x^*

2.6 Mixed Integer Linear Programming

Mixed-Integer Linear Programming (MILP) is a mathematical method that aims to minimize or maximize a linear function within a defined subset of $\mathbb{R}^n$, The subset is

shaped by a system of linear equalities and inequalities, with the distinct requirement that some of the decision variables take integer values, while others can remain continuous.

MILP originates from Linear Programming (LP), which focuses purely on continuous variables. While LP allows for optimization within linear constraints, it couldn't address scenarios requiring discrete decisions, such as determining whether to build a factory (binary choice: yes or no) or allocating whole units of a resource (integer quantities). Recognizing this gap, MILP was introduced to incorporate integrality constraints, expanding its applicability to more complex, real-world problems.

Since its formalization in the mid-1960s, MILP has evolved into a sophisticated and widely used optimization tool. Advances in computational algorithms, such as branch-and-bound and branch-and-cut, have significantly enhanced its efficiency and scalability. Today, MILP has become a cornerstone of operations research, supported by robust commercial software packages like CPLEX [29], Gurobi [25], and FICO Xpress [19]. These tools allow practitioners to model and solve large-scale optimization problems with precision, reliability, and speed.

The versatility of MILP is evident in its extensive applications across industries, including logistics, manufacturing, finance, and energy systems [30]. By blending theoretical elegance with practical utility, MILP enables decision-makers to find optimal solutions to challenges involving both discrete and continuous variables.

2.6.1 General Form of MILP

As mentioned above (in ILP), MILP is the general form when some variables are restricted to integers and others are continuous.

$$\text{Minimize or Maximize } Z = c_x^T x + c_y^T y \tag{2.14}$$

Subject to:

$$A_x x + A_y y \leq b \tag{2.15}$$

$$D_x x + D_y y = e \tag{2.16}$$

$$x \geq 0, \quad x \in \mathbb{R}^n \tag{2.17}$$

$$y \geq 0, \quad y \in \mathbb{Z}^p \tag{2.18}$$

where:

- x is the vector of continuous decision variables.
- y is the vector of integer decision variables.
- c_x, c_y are coefficient vectors for the objective function.
- A_x, A_y, D_x, D_y are matrices of coefficients for the constraints.
- b, e are vectors of constants.

MILP is inherently a non-convex optimization problem due to the presence of integer variables, which introduce discrete decision spaces. In contrast, if all variables are required to be integers ($p = n$), the problem becomes a purely Integer Linear Programming (ILP) problem, where any feasible solution is a completely integer vector. Even in the more general case where $p \leq n$, we define an integer solution to (1.7)–(1.9) as a vector whose last components p are integer values, while the others may remain continuous.

2.6.2 MILP Solution Techniques

Mixed-Integer Linear Programming (MILP) problems involve both continuous and integer decision variables with linear objectives and constraints. Due to the combinatorial complexity introduced by integer variables, exact solving methods are required. Modern MILP solvers like Gurobi [25], CPLEX [29], and HiGHS [27] implement a combination of foundational algorithms enhanced with heuristic and machine learning techniques to improve performance.

1- Branch and Bound Method:

The Branch and Bound (B&B) algorithm is a widely used method for solving Mixed-Integer Linear Programming (MILP) problems. It combines **relaxation**, **branching**, and **bounding** to systematically explore the feasible solution space. At each step, the algorithm:

- Solves a relaxed version of the MILP (without integrality constraints).
- If the solution is integer feasible, it updates the best known solution.
- If not, it selects a variable with a fractional value and branches on it, creating two new subproblems with additional constraints.

- Subproblems whose relaxed solution cannot improve the current best solution are discarded (pruned).

The process continues until all nodes are either pruned or solved, ensuring that the optimal integer solution is found.

Algorithm 2 Branch and Bound Algorithm for MILP

```
Initialize best_solution ← ∞
Initialize queue with root node
while queue not empty do
    current_node ← SelectNode(queue)
    Solve LP relaxation at current node
    z ← objective value, x* ← solution
    if z ≥ best_solution then
        Prune node
    else if x* is integer feasible then
        Update best_solution ← min(z, best_solution)
    else
        Select fractional variable x_i
        Create left node: x_i ≤ ⌊x_i*⌋
        Create right node: x_i ≥ ⌈x_i*⌉
        Add both nodes to queue
    end if
end while
return best_solution
```

where z is the variable that holds the optimal solution value (in this case, for the relaxation problem) in each iteration of the algorithm, x^* is the solution to the relaxation or linearized problem at the current node.

2- Branch and Cut Algorithm for MILP:

The Branch and Cut algorithm is a robust and widely used method for solving Mixed-Integer Linear Programming (MILP) problems. It extends the classical Branch and Bound framework by incorporating cutting-plane techniques to tighten the linear programming (LP) relaxation and reduce the search space [31].

At each node of the search tree, the algorithm solves the LP relaxation. If the solution is not integer-feasible, it attempts to generate cutting planes to eliminate the fractional solution. If no violated cuts are found, the node is branched on a selected variable. This combination of bounding, cutting, and branching allows the method to efficiently explore and prune the solution space. For a detailed structure, see Algorithm 3.

Algorithm 3 Branch and Cut Algorithm for MILP

```
Initialize best_solution ← ∞                                   ▷ For minimization
Initialize queue with the root node
while queue not empty do
    current_node ← select node from queue
    Solve LP relaxation at current node to obtain z, x*
    if z ≥ best_solution then
        Prune current node
    else
        if x* is integer feasible then
            Update best_solution ← min(z, best_solution)
        else
            Choose variable x_i with fractional value in x*
            Create left node with constraint x_i ≤ ⌊x_i*⌋
            Create right node with constraint x_i ≥ ⌈x_i*⌉
            Add both nodes to queue
        end if
    end if
end while
return best_solution
```

where:

x^* Solution vector of the LP relaxation at the current node.

z Objective value corresponding to x^*.

x_i Fractional variable selected for branching.

best_solution Best (lowest) objective value found among integer-feasible solutions.

Queue Set of active subproblems to be processed.

3- Heuristics:

Solving Mixed-Integer Linear Programming (MILP) problems exactly can be computationally challenging, especially for large instances. Heuristic methods provide practical alternatives by quickly finding high-quality feasible solutions without guaranteeing optimality. Among these, Local Branching and Relaxation Induced Neighborhood Search (RINS) are two effective techniques widely used in practice [20].

- **Local Branching:** This heuristic explores solutions near a given incumbent by restricting the number of binary variable changes in a neighborhood, enabling faster improvement searches [20].

- **Relaxation Induced Neighborhood Search (RINS):** RINS fixes variables that have the same values in both the incumbent and LP relaxation solutions and solves the reduced MILP to find better solutions efficiently [20].

2.7 Mixed Integer Non-Linear Programming

Mixed-Integer Nonlinear Programming (MINLP) represents a robust optimization framework that simultaneously addresses both continuous and discrete variables while considering nonlinear constraints or objective functions. This versatile approach has garnered substantial attention due to its capacity to model intricate real-world challenges across diverse domains, including engineering design, operations research, and robotics.

MINLP integrates the complexities of combinatorial and nonlinear optimization, rendering it an inherently challenging problem class that necessitates the continuous development of innovative algorithms and specialized software to efficiently address large-scale instances. Among the prevalent methods devised for solving MINLP problems are the branch-and-bound algorithm, outer approximation techniques, and hybrid approaches. These methodologies frequently leverage advancements in mixed-integer linear programming (MILP) and nonlinear programming (NLP), significantly enhancing their computational efficacy.

Mixed-Integer Nonlinear Programming (MINLP) is generally considered an intractable problem class due to the combination of discrete and continuous decision variables. As noted by Belotti et al[9]. MINLP encompasses both decidable and undecidable subproblems depending on the structure of the constraints and objective function . In particular, problems involving nonconvex functions or unbounded domains may lead to formulations that are formally undecidable — meaning no algorithm can guarantee a solution in finite time — whereas convex or bounded cases tend to remain within decidable complexity classes. To address this challenge, various heuristic and approximation strategies have been developed, including the Feasibility Pump [12], diving algorithms [14], and Relaxation Induced Neighborhood Search (RINS) [22], many of which have been adapted from MILP and customized for MINLP settings.

The evolving landscape of MINLP research continues to yield advancements in algorithms, computational tools, and practical applications. Emerging techniques, such as the Extended Supporting Hyperplane (ESH) algorithm, demonstrate promise in addressing convex MINLP problems with improved efficiency [39]. As the demand for addressing complex optimization scenarios increases across various industries, the role

of MINLP methodologies is anticipated to expand, offering innovative solutions to increasingly sophisticated real-world challenges.

2.7.1 General form of MINLP

Mixed-Integer Nonlinear Programming combines elements of NLP and ILP, involving both continuous and integer variables, and at least one nonlinear function in the objective or constraints.

$$\text{Minimize or Maximize } f(x,y) \tag{2.19}$$

Subject to:

$$g_i(x,y) \leq 0 \quad \text{for } i = 1,\ldots,m_1 \tag{2.20}$$

$$h_j(x,y) = 0 \quad \text{for } j = 1,\ldots,m_2 \tag{2.21}$$

$$x \in X \subseteq \mathbb{R}^n \tag{2.22}$$

$$y \in Y \subseteq \mathbb{Z}^p \tag{2.23}$$

where:

- x is the vector of continuous decision variables.
- y is the vector of integer decision variables.
- $f(x,y)$: the objective function to be minimized or maximized,
- $g_i(x,y) \leq 0$: the set of inequality constraints,
- $h_j(x,y) = 0$: the set of equality constraints,

If at least one of the following components is nonlinear—the objective function $f(x,y)$, any inequality constraint $g_i(x,y)$, or any equality constraint $h_j(x,y)$—the problem is classified as a Mixed-Integer Nonlinear Programming (MINLP) problem.

MINLP problems are among the most difficult to solve, as they combine the complexities of nonlinearity with the combinatorial nature of integer variables. Like nonlinear programming (NLP) problems, MINLPs can be either convex or non-convex depending on the characteristics of the functions f, g, and h.

2.7.2 Solving MINLP

- **NLP Branch and Bound for MINLP:**

  NLP Branch and Bound is an exact global optimization algorithm designed to solve non-convex Mixed-Integer Nonlinear Programming (MINLP) problems. It extends the classical Branch and Bound approach by using Nonlinear Programming (NLP) relaxations at each node instead of linear relaxations.

  This method recursively partitions the feasible space defined by integer variables and uses NLP solvers to compute lower bounds. These bounds help eliminate regions that cannot contain the optimal solution (pruning), while feasible integer solutions help improve the global upper bound [13].Refer to Algorithm 4 for further illustration of the method.

Algorithm 4 NLP Branch and Bound for MINLP [13]

Initialize: Create list $L \leftarrow \{(\ell^I, u^I)\}$, set upper bound $z_U \leftarrow \infty$, best solution $x^* \leftarrow$ NONE
while $L \neq \emptyset$ **do**
  Select and remove a problem $N_i = (\ell_i^I, u_i^I)$ from L
  Solve NLP relaxation $\text{NLP}_R(\ell_i^I, u_i^I)$
  if infeasible **then**
    Continue to next node
  end if
  Let $\hat{x}_i$ be solution, and $z_i = \text{objective}(\hat{x}_i)$
  if $z_i \geq z_U$ **then**
    Prune node
  else if $\hat{x}_i$ is integer feasible **then**
    Update $z_U \leftarrow z_i$, $x^* \leftarrow \hat{x}_i$
    Remove from L all nodes with lower bound $\geq z_U$
  else
    Branch: Select a fractional variable and divide region into subproblems
    Add new subproblems to L
  end if
end while
return x^*

where:

- $\hat{x}_i$: solution to the NLP relaxation at node N_i.
- z_i: objective value corresponding to $\hat{x}_i$.
- z_U: current best known upper bound (from integer-feasible solutions).
- $\mathcal{L}$: list of active subproblems (nodes).

– (ℓ_i^I, u_i^I): bounds on integer variables defining node N_i.

- **Extended Cutting Plane (ECP) Method:**

  The Extended Cutting Plane (ECP) method is a widely used algorithm designed to solve convex Mixed-Integer Nonlinear Programming (MINLP) problems. Unlike the Outer Approximation (OA) method, which requires solving both NLP and MILP subproblems, ECP focuses solely on solving a series of MILP relaxations. This makes it computationally attractive, especially for large-scale convex problems[13].

  At each iteration, the algorithm linearizes the nonlinear constraints around the current solution point and adds these linear approximations (cuts) to the MILP model. By successively tightening the feasible region through these cuts, the solution gradually approaches the feasible region of the original nonlinear problem.

  Although ECP avoids solving expensive NLP subproblems, it generally requires more iterations to converge compared to OA or NLP-based approaches. However, its reliance on linear programming makes it scalable and robust in many practical settings.

Algorithm 5 Extended Cutting Plane (ECP) for MINLP [13]

Initialize: Choose initial feasible point (x^0, y^0); set $k = 0$.
repeat
    Linearize nonlinear constraints at (x^k, y^k).
    Solve MILP relaxation.
    if solution violates any nonlinear constraint **then**
        Add corresponding linearizations as cuts.
    end if
    Update solution to (x^{k+1}, y^{k+1}).
until all nonlinear constraints are satisfied
return Feasible solution satisfying all constraints.

- **Outer Approximation (OA):** Outer Approximation is an iterative algorithm designed for convex MINLP problems. It alternates between solving a nonlinear programming (NLP) subproblem and a mixed-integer linear programming (MILP) master problem. In each iteration, the NLP subproblem is solved to obtain a feasible solution, and linear approximations (cuts) of the nonlinear constraints are added to the MILP master problem to refine the search space. This process continues until convergence to an optimal solution .[13]

- **Neural Network Methods for Solving MINLP:**

  With the increasing complexity of Mixed-Integer Nonlinear Programming (MINLP) problems [9], neural networks have emerged as powerful tools to assist traditional solvers. Deep learning and graph-based models help handle non-convex and combinatorial aspects by:

  - Predicting branching or node priorities in Branch-and-Bound.
  - Approximating objectives or feasible regions.
  - Guiding heuristics or cut generation.
  - Reducing runtime by pruning poor regions.

  Though not always exact, these models can significantly speed up the computational time, especially when many similar instances are solved repeatedly. This reflects the rise of *learning-augmented optimization* [13].

- **Modern developments related to MINLP:**

  In recent years, significant advancements have been made in solving Mixed-Integer Nonlinear Programming (MINLP) problems, with the development of modern algorithms that extend beyond classical methods such as Branch and Bound and Outer Approximation. These contemporary approaches aim to enhance computational efficiency, scalability, and robustness, particularly when addressing large-scale, non-convex, or highly constrained problems encountered in real-world applications[10].

  - **Global Optimization-Based Solvers:** Modern solvers like `BARON`, `ANTIGONE`, and `SCIP` use global optimization techniques, including spatial Branch and Bound, convex relaxations, and cutting-plane generation to find global optima even for non-convex MINLPs.
  - **Decomposition Methods:** Techniques like Benders Decomposition and Generalized Benders Decomposition decompose the problem into master and subproblems, enabling parallel processing and better handling of complicating variables.
  - **Surrogate and Metaheuristic Approaches:** Surrogate models (e.g., Gaussian processes or neural networks) approximate objective and constraint

functions to reduce evaluation costs. Metaheuristics such as Genetic Algorithms, Particle Swarm Optimization, and Simulated Annealing are also employed when optimality guarantees can be relaxed.

- **Learning-Augmented Optimization:** Machine learning models, including neural networks and graph neural networks (GNNs), are increasingly integrated into solvers to predict branching decisions, generate feasible solutions, or approximate relaxations. This hybridization improves performance on specific problem classes.

- **Solver Frameworks with Modular Architectures:** Recent frameworks like `Pyomo` [44], `JuMP`, and `MINLPy` support customization and integration of classical and ML-based methods, allowing researchers to design and experiment with new strategies quickly.

  These modern approaches are particularly useful for real-time applications, black-box optimization problems, or domains where exact methods become computationally prohibitive.

2.8 Problem Complexity and Classification

In computational complexity theory, problems are categorized according to the computational resources required to solve them. These resources primarily include time and memory [4]. The most commonly discussed complexity classes are:

Class P (Polynomial Time)

Class **P** includes all decision problems that can be solved by a deterministic Turing machine in polynomial time. These problems are considered "tractable" or efficiently solvable.

Class NP (Nondeterministic Polynomial Time)

Class **NP** includes all decision problems for which a proposed solution can be verified in polynomial time by a deterministic Turing machine, even if finding the solution itself may not be feasible in polynomial time.

NP-Complete Problems

A problem is said to be **NP-complete** if:

- It belongs to NP.
- Every problem in NP can be reduced to it in polynomial time.

NP-complete problems are the hardest problems in NP. Solving any one of them in polynomial time would imply **P = NP**.

NP-Hard Problems

NP-hard problems are at least as hard as the hardest problems in NP but may not belong to NP themselves. They might not even be decision problems or verifiable in polynomial time.

Does P = NP?

One of the most important open problems in theoretical computer science is:

Is P equal to NP?

This question asks whether every problem whose solution can be verified quickly can also be solved quickly. It remains unresolved and is one of the seven Millennium Prize Problems. And no polynomial-time algorithm has yet been found to solve them—unless, as the hypothesis suggests, P = NP. Due to the importance of solving such problems, scientists continue to develop approximation, heuristic, and metaheuristic methods to tackle them. (See figure 2.1).

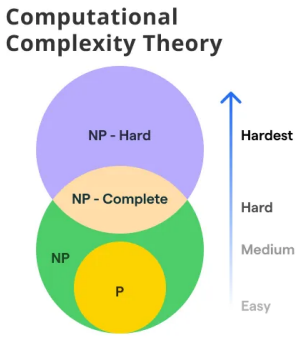


Figure 2.1: Visual representation of complexity classes: P, NP, NP-Complete, and NP-Hard

2.9 Comparison between Optimization models

Table 2.1 presents a comprehensive comparison of major mathematical optimization paradigms used in operations research and computational optimization. The comparison spans from Linear Programming (LP), the most basic and computationally tractable form, to the highly complex Mixed Integer Nonlinear Programming (MINLP) that combines both integer variables and nonlinear functions. Each model represents a different trade-off between expressiveness and computational efficiency, with increasing modeling power generally coming at the cost of solution difficulty. This systematic comparison highlights the key characteristics that differentiate these optimization approaches and guides practitioners in selecting the appropriate model for specific problem domains.

Table 2.1: Comparison of Optimization Models

Feature	Linear Programming (LP)	Nonlinear Programming (NLP)	Integer Linear Programming (ILP)	Mixed Integer Linear Programming (MILP)	Mixed Integer Nonlinear Programming (MINLP)
Decision Variables	Continuous	Continuous	Integer	Mix of integer and continuous	Mix of integer and continuous
Objective Function	Linear	Nonlinear	Linear	Linear	Nonlinear
Constraints	Linear	Can be nonlinear	Linear	Linear	Can be nonlinear
Complexity	Polynomial time (easy)	NP-hard in general	NP-hard	NP-hard	NP-hard
Possible Solution Methods	Simplex, Graphical method	Gradient descent, Interior point methods	Branch and bound, Cutting plane	Branch and bound, Branch and cut	Outer approximation, NLP Branch&Bound, Neural Network
Typical Solve Time	Fast	Moderate to slow (depends on problem structure)	Slow (problem size dependent)	Slow (problem size dependent)	Very slow
Global Optimality	Always guaranteed	Only for convex problems	Guaranteed (if solved to completion)	Guaranteed (if solved to completion)	Only in special cases
Common Applications	Resource allocation, Portfolio optimization, Production planning	Engineering design, Process control, Machine learning	Scheduling, Assignment problems, Facility location	Supply chain, Network design, Production planning with setup times	Process synthesis, Engineering design with discrete choices
Software Tools	CPLEX, Gurobi, GLPK	IPOPT, SNOPT, KNITRO	CPLEX, Gurobi, CBC	CPLEX, Gurobi, SCIP	BARON, Couenne, DICOPT
Strengths	Efficient algorithms, Fast solution, Widely available solvers	Can model nonlinear relationships, More realistic in many applications	Can model indivisible resources or yes/no decisions	Combines power of integer and continuous variables	Most general form, can handle complex real-world systems
Limitations	Cannot handle nonlinear relationships, integer variables	May converge to local optima, Harder to solve	Limited to linear relationships	Limited to linear relationships, Solution time grows with integer variables	Very difficult to solve, Often requires problem-specific decomposition
Convexity	Always convex	Can be convex or non-convex	Discrete (non-convex)	Discrete (non-convex)	Usually non-convex

2.10 Conclusion

This chapter traced the evolution of optimization models from the linear and continuous world of LP to the expressive, hybrid nature of MINLP. Each advancement in model formulation—NLP, ILP, MILP, and finally MINLP—has enabled practitioners to address increasingly complex decision problems involving both discrete and nonlinear elements. These techniques not only offer richer modeling capacity but also present new computational challenges, particularly in terms of scalability and tractability.

As highlighted by Grossmann [23], the development of global optimization algorithms such as branch-and-bound, outer approximation, and generalized disjunctive programming has significantly improved the solvability of large-scale MINLP problems. Moreover, the emergence of hybrid heuristics and decomposition strategies has made these methods more applicable to real-time and industrial environments.

The insights gained from this chapter form the theoretical bedrock for the modeling choices and solver selection presented in subsequent parts of this thesis. Specifically, they justify the use of MINLP and the adoption of the SCIP solver for solving Cut Order Planning problems—a context where complex combinatorics and nonlinear constraints must be addressed simultaneously.

CHAPTER 3

MATHEMATICAL MODELING OF THE PROBLEM

3.1 Introduction

This chapter presents the mathematical formulation used to solve the Cut Order Planning (COP) problem in the apparel industry, where efficient fabric utilization is critical due to its significant contribution—typically 50–60%—to total production costs. The model aims to determine the most efficient way to allocate and cut fabric to fulfill varying customer demands across multiple garment sizes and styles.

The formulation used in this study is based on the work of Ünal and Yüksel [48], who proposed a Mixed-Integer Nonlinear Programming (MINLP) approach to COP in a mid-sized apparel company. Their model captures key operational aspects such as fabric spreading limits, marker lengths, and demand fulfillment while minimizing fabric waste. It integrates both discrete decisions (e.g., number of plies, garment counts) and continuous ones (e.g., marker lengths), making it suitable for real-world garment production environments.

In this chapter, we present the model reimplementation using the open-source SCIP solver through the Pyomo modeling library in Python. Compared to the original LINGO-based formulation, this implementation is more transparent, reproducible, and accessible—especially for small and medium-sized enterprises seeking cost-effective optimization solutions.

The chapter is organized as follows:

- Problem context and key production constraints,
- Definition of sets, parameters, and decision variables,

- Mathematical formulation of the objective function and constraints,
- Discussion of model complexity and classification as MINLP,
- Feasibility justification and real-world applicability,
- Optimization workflow and execution environment.

This model forms the foundation for the case study presented in Chapter **??**, where it is tested on real production data from the apparel industry and compared to both LINGO and other open-source solvers.

3.2 Problem Context

In garment manufacturing, the cutting department is tasked with converting raw fabric into cut components that fulfill a variety of customer orders. In a typical apparel factory, fabrics arrive in bulk rolls and are processed in the cutting department according to pre-planned marker layouts. These layouts dictate how different sizes and styles are arranged across layers of fabric. The key constraints include the maximum length and width of fabric lays, the grouping of sizes for production efficiency, and the balancing of order fulfillment across order items.

The challenge in COP lies in the trade-off between fabric efficiency and operational feasibility. For instance, combining too many size-color-style combinations in a single marker may increase fabric efficiency but make spreading and cutting operations more difficult. Conversely, creating separate markers for each SKU is operationally easier but highly wasteful.

This operational modeling aligns with the objectives defined in Chapter 1, particularly in addressing fabric utilization through advanced mathematical techniques.

3.3 Mathematical Model Formulation

The objective of the Cut Order Planning (COP) model is to determine an optimal combination of garments across markers and fabric lays to minimize total fabric consumption while meeting all customer demands. The model captures the practical constraints of fabric spreading and cutting operations typically encountered in apparel production.

The full mathematical formulation is presented below:

Sets and Indices

- $i \in I$: Set of spreadings (layers), $i = \{1, 2, \ldots, n\}$
- $j \in J$: Set of garment sizes, $j = \{1, 2, \ldots, m\}$

Parameters

- S_j: Order quantity for size j (pieces)
- B_j: Estimated length per piece of size j in marker plan (meters)
- M: Cutting table length (meters)
- $K_{\max}$: Maximum piles allowed per spreading (integer)
- P: Allowed excess cutting rate (ECR) (%)

Decision Variables

- K_i: Number of piles in spreading i (integer)
- A_{ij}: Number of pieces of size j in each pile of spreading i (integer)
- T_i: Length of marker plan for spreading i (continuous)

Objective Function

Minimize the total fabric usage:

$$\text{Minimize } \sum_{i=1}^{n} K_i \cdot T_i$$

Constraints

1. The number of plies for each spreading must not exceed the maximum allowed:

$$K_i - K_{\max} \leq 0 \quad \forall i = 1, 2, \ldots, n$$

2. The total number of cut pieces must be at least equal to the order quantity for each size:

$$\sum_{i=1}^{n} K_i \cdot A_{ij} \geq S_j \quad \forall j = 1, 2, \ldots, m$$

3. The total number of cut pieces must not exceed the order quantity plus the excess

cutting share:

$$\sum_{i=1}^{n} K_i \cdot A_{ij} \leq S_j \left(1 + \frac{P}{100}\right) \quad \forall j = 1, 2, \ldots, m$$

4. The length of the marker plan must be shorter than the table length for any spreading:

$$\sum_{j=1}^{m} A_{ij} \cdot B_j \leq M \quad \forall i = 1, 2, \ldots, n$$

5. The number of plies decreases as the spreading index increases:

$$K_i - K_j \geq 0 \quad \forall i > j$$

6. The length of each spreading is determined by the marker plan:

$$\sum_{j=1}^{m} A_{ij} \cdot B_j - T_i = 0 \quad \forall i = 1, 2, \ldots, n$$

7. All variables are non-negative:

$$A_{ij}, K_i, T_i, S_j, B_j, K_{\max}, P, M \geq 0$$

This model effectively balances precision and computational tractability while addressing key operational constraints in cut order planning. It captures critical relationships between garment sizes, marker layouts, fabric usage, ply counts, and spreading length, all of which are essential for minimizing total fabric consumption. Given the combinatorial nature of size assignment and the nonlinear dependencies in fabric length calculations, the formulation justifies the adoption of a Mixed Integer Nonlinear Programming (MINLP) approach.

3.4 Model Type and Complexity

The Cut Order Planning (COP) problem formulated in this thesis is classified as a **Mixed-Integer Nonlinear Programming (MINLP)** problem, based on two defining characteristics:

- **Nonlinearity**: The model includes bilinear terms such as $K_i \cdot A_{ij}$, which appear in constraints governing the number of cut pieces and the marker length calcu-

lation. These expressions represent real-world relationships where the number of piles and the number of pieces per size jointly determine fabric usage. Additionally, constraints like $\sum_j A_{ij} \cdot B_j = T_i$ introduce dependencies between decision variables, making the model nonlinear.

- **Mixed-Integer Nature**: The decision variables A_{ij} and K_i are integer-valued, reflecting quantities such as garment counts and pile numbers, while T_i is a continuous variable representing marker lengths. This combination of discrete and continuous decisions is typical of MINLP models.

From a computational standpoint, COP is an NP-hard problem. The COP problem belongs to the class of NP-hard problems because it involves both discrete decisions and nonlinear constraints, and it generalizes the classical cutting stock and bin packing problems. As a result, its solution space grows exponentially, making it computationally intractable for large instances without advanced optimization techniques. The search space expands combinatorially with each additional size or layer, making brute-force enumeration infeasible. Efficient solving requires specialized algorithms capable of navigating this large, complex space [40].

Given the complexity of the model, careful attention must be paid to instance sizing, data structuring, and constraint formulation. In real-world applications, it may be beneficial to use hybrid approaches that combine exact optimization with heuristics to handle large-scale problems effectively.

3.5 Feasibility and Justification

The feasibility of the COP model has been validated through logical consistency checks, constraint verification, and comparison with real-world data.

Validation of Model Logic

The model ensures that:

- All constraints — including ply limits, table length, and excess cutting rate — are respected,
- Total cut pieces meet or slightly exceed demand within an acceptable margin,
- Marker lengths do not exceed available table length,
- Ply counts decrease with spreading index to avoid inefficiencies.

Logical verification shows that the model behaves correctly under various test cases and adheres to the principles of efficient fabric utilization.

Reasonableness of Assumptions

Key assumptions made in the model include:

- Fixed order quantities — common in mass production settings where orders are confirmed before cutting begins.

- Uniform fabric width — applicable when working with consistent rolls; can be extended to variable widths in future work.

- Constant garment sizes — minor variations exist but do not significantly affect layout efficiency.

- Deterministic Excess Cutting Rate (ECR) — practical for planning purposes, though probabilistic values could be used in highly dynamic environments.

These assumptions simplify computation and are valid for most batch-style apparel production scenarios.

Real-World Alignment

The model has been tested using datasets derived from two real-world apparel companies producing garments such as shirts, coats, trousers, and sweatshirts. Results show that:

- The model consistently reduces fabric usage compared to manual planning (average saving: **7%**, up to **13%** in some cases),

- Longer marker plans lead to better fabric utilization — confirming an important insight from industry practice,

- Using open-source solvers like `SCIP` achieves similar or better performance than proprietary tools like `LINGO`, with added benefits in transparency and adaptability.

These findings demonstrate that the model is not only mathematically valid but also operationally relevant and industrially applicable.

Computational Efficiency and Practical Applicability

By reimplementing the original LINGO-based MINLP model in Python using `Pyomo` and `SCIP`, we achieved:

- Faster solution times.

- Greater flexibility in modifying input parameters and constraints.

- Improved reproducibility for researchers and practitioners.

This confirms the suitability of the model for industrial use and supports the thesis's broader objective of promoting open-source optimization tools in small-to-medium enterprises.

3.6 Implementation Framework and Solver Environment

3.6.1 Optimization Workflow Overview

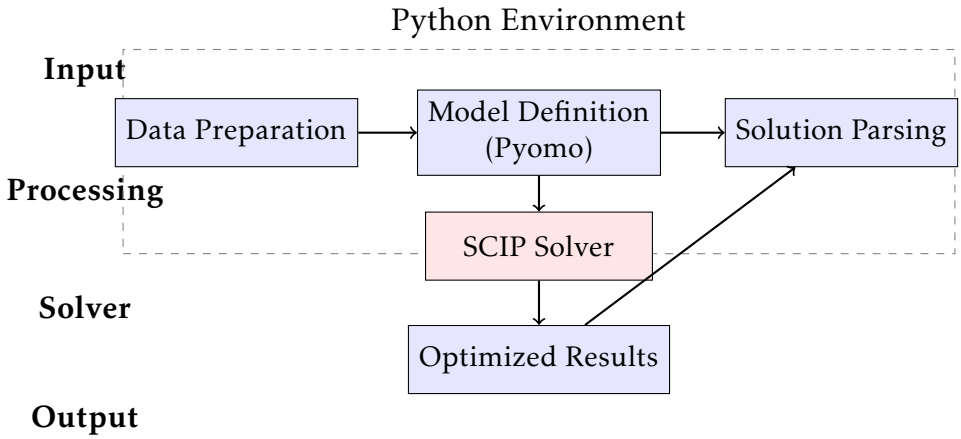


Figure 3.1: Flowchart of Optimization Framework Using Python and SCIP

Figure 3.1 illustrates the overall workflow for solving the Cut Order Planning (COP) problem using Python and the SCIP solver. The process begins with data preparation, where order quantities and garment specifications are structured as input parameters. This is followed by model definition using the Pyomo modeling framework, which encodes the objective function and all constraints. The formulated problem is then passed to the SCIP solver, which executes the optimization process. Upon completion, solution parsing is performed to extract fabric usage, spreading configurations, and other decision variables. The final results are made available for reporting and further analysis. The modular structure allows easy experimentation and solver replacement if needed.

For a detailed step-by-step example demonstrating the implementation of this optimization workflow using Pyomo and SCIP, refer to **Appendix A**.

3.6.2 Model Execution Environment

The COP model was implemented using Python, an open-source high-level programming language widely adopted in scientific computing and data analysis due to its readability and extensive ecosystem of libraries.

To model the optimization problem, we used **Pyomo**, an open-source Python-based algebraic modeling language that supports the formulation and analysis of linear, nonlinear, and mixed-integer mathematical programming problems [44]. Pyomo provides a high-level syntax for defining optimization variables, constraints, and objectives in a way that closely resembles the mathematical formulation.

To solve the model, we used **SCIP** (Solving Constraint Integer Programs), one of the most advanced non-commercial solvers for mixed-integer nonlinear programming (MINLP). Developed by the Zuse Institute Berlin, SCIP integrates constraint programming, branch-and-bound, and presolving techniques to handle nonconvex and discrete optimization problems efficiently [1].

The SCIP solver (version 9.2.2) was integrated via Pyomo's `SolverFactory` interface and executed locally as shown below:

```
solver = SolverFactory('scip',
          executable='/home/lenovo/scipoptsuite-9.2.2/build/bin/scip')
```

All experiments were executed using the same data structure described earlier to ensure fair comparison between solvers. No solver parameter tuning was applied, in order to preserve default behavior and isolate solver performance from manual configuration.

Experiments were conducted in a Python 3.12.3 virtual environment under the Windows Subsystem for Linux (WSL), running Ubuntu. The Python interpreter path was:

```
/home/lenovo/venv/scip-env/bin/python
```

Model development, testing, and debugging were carried out using **PyCharm 2025.1.1 Academic Edition**, an integrated development environment (IDE) developed by JetBrains and widely used in scientific and industrial Python projects. PyCharm offers features such as intelligent code completion, syntax highlighting, environment management, and built-in terminal support, which facilitate efficient and organized model implementation.

3.7 Conclusion

This chapter introduced a mathematical model for the Cut Order Planning (COP) problem in the apparel industry, structured as a Mixed-Integer Nonlinear Programming (MINLP) formulation. The model, originally proposed by Ünal and Yüksel [48], addresses the trade-off between fabric efficiency and production feasibility by modeling real-world constraints such as table length, ply limits, and marker layouts.

We reimplemented the model using the Pyomo library and SCIP solver in Python, providing an open-source alternative to the original LINGO-based implementation. This adaptation enhances transparency, flexibility, and reproducibility—essential qualities for practical adoption by small and medium-sized apparel manufacturers.

The model's NP-hard classification was discussed in relation to its nonlinear and combinatorial structure, reflecting the underlying complexity of the problem. The formulation's validity and feasibility were supported through constraint verification, assumption analysis, and alignment with industry practices.

Ultimately, this model provides a robust and scalable framework for optimizing fabric consumption in garment production. The next chapter builds upon this work by applying the model to real-world apparel datasets and analyzing the resulting performance in comparison to alternative solvers.

CHAPTER 4
CASE STUDY

4.1 Introduction

The primary goal of this chapter is to identify a computationally efficient solution strategy for solving the Cut Order Planning (COP) model introduced in Chapter 3. While several solvers are available for MINLP problems, their performance can vary significantly depending on problem size and structure. In this work, we test multiple open-source solvers, including Couenne, Bonmin, and SCIP, with a particular focus on their ability to solve the benchmark coat production case from Ünal and Yüksel.

Preliminary experiments revealed that both Couenne and Bonmin failed to produce a solution for the coat production instance, even after extended runtimes exceeding two hours. This confirms the need for a more robust and efficient solver. SCIP, on the other hand, was able to provide a feasible solution in significantly less time, making it the preferred solver for our implementation. The rest of this chapter outlines the data structure, software environment, implementation approach, and validation strategy using SCIP.

4.2 Description Of The Case Study:

The case study focuses on a mid-sized apparel manufacturing company located in a textile production hub in Turkey, as described in the study by Ünal and Yüksel [48]. This company specializes in producing both basic and fashion-oriented garments such as shirts, trousers, coats, and sweatshirts. With approximately 450 employees, the company operates under a make-to-order production strategy, aiming to meet domes-

tic and international demand with flexibility and minimal inventory. It is equipped with a dedicated cutting department, which plays a critical role in overall fabric efficiency. Improving operations within this department—particularly through optimized Cut Order Planning (COP)—has a direct impact on production costs, as fabric typically accounts for 50–60% of total manufacturing expenses.

The primary objective is to enhance operational efficiency in the cutting department, recognized as one of the most fabric-intensive stages of production. As fabric costs continue to escalate, optimizing material utilization has become critical.

Currently, the company employs manual planning methods supported by basic spreadsheet tools. This approach relies heavily on the expertise of cutting room supervisors, often resulting in fabric over-consumption due to suboptimal spreading and cutting schedules. Additionally, managing multiple garment sizes per order introduces complexity to manual planning, increasing risks of exceeding order tolerances or material waste.

Key operational challenges include:

- Balancing multiple garment sizes within single markers
- Adhering to maximum table lengths and fabric ply limitations
- Meeting strict delivery timelines
- Minimizing fabric waste while preventing garment shortages

To address these challenges, Ünal and Yüksel has adopted a mixed-integer nonlinear programming (MINLP) framework using the LINGO solver to optimize its Cut Order Planning (COP) process. In this work, we aim to replace it with a modern SCIP-based optimization framework implemented in Python, with the objective of improving solver efficiency and flexibility.

4.3 Input Data Used:

The input parameters used in this study cover four product categories: shirts, coats, trousers, and sweatshirts. These are summarized in the following tables, which detail the order quantities and estimated fabric lengths required for each size.

All input data were adapted directly from the original Cut Order Planning case study presented by Ünal and Yüksel [48].

Table 4.1: Input Data for Shirt Production

Size	Sj=Order Quantity (pieces)	Bj=Estimated Length (cm)
T39	69	127.28
T41	96	127.28
T43	90	127.28
T45	45	127.28

Table 4.2: Input Data for Coat Production

Size	Sj=Order Quantity (pieces)	Bj=Estimated Length (cm)
T48	37	205.49
T50	84	205.49
T52	84	205.49
T54	84	205.49
T56	59	205.49
T58	38	205.49
T60	22	205.49
T62	8	205.49

Table 4.3: Input Data for Trouser Production

Size	Sj=Order Quantity (pieces)	Bj=Estimated Length (cm)
T24	40	112.22
T26	205	112.22
T28	35	112.22
T30	10	112.22
T32	10	112.22

Table 4.4: Input Data for Sweatshirt Production

Size	Sj=Order Quantity (pieces)	Bj=Estimated Length (cm)
S	130	100
M	348	100
L	444	100
XL	304	100
XXL	152	100

4.4 Results and Discussion

4.4.1 SCIP vs. LINGO: Comparative Analysis Across Products

This subsection presents a detailed comparison between the results obtained from SCIP and the original LINGO-based implementation from Ünal and Yüksel across

four COP scenarios: shirt, coat, trouser, and sweatshirt. The comparison focuses on total fabric usage, number of spreadings, and solution efficiency (measured by solver iterations). The aim is to assess whether the open-source SCIP solver can match or outperform the commercial LINGO environment in terms of both optimality and computational practicality.

4.4.1.1 Shirt Production: SCIP vs. LINGO

The shirt production case involves four size categories (T39, T41, T43, T45), with a maximum table length of 1600 cm and a fabric ply capacity of up to 200 layers. The average fabric length per piece is 127.28 cm. The goal is to fulfill demand while minimizing total fabric usage.

Note: The maximum number of plies used in this case was $K_{\max} = 200$.

Table 4.5: LINGO Result – Shirt Case (Ünal & Yüksel)

Sizes	**T39**	**T41**	**T43**	**T45**	**No. of piles**	**Marker plan length(cm)**
Total order	69	96	90	45		
1. Spreading	2	2	3	0	30	891.01*
2. Spreading	1	4	0	5	9	1272.8*
No. of pieces	69	96	90	45	**Total length, cm**	
EQR, %	0	0	0	0	**38752.32**	

Table 4.6: SCIP Result – Shirt Case

Sizes	**T39**	**T41**	**T43**	**T45**	**No. of piles**	**Marker plan length(cm)**
Total order	69	96	90	45		
1. Spreading	3	0	6	3	15	1527.36
2. Spreading	2	8	0	0	12	1272.80
No. of pieces	69	96	90	45	**Total length, cm**	
EQR, %	0	0	0	0	**38184.00**	

As shown in Tables 4.6 and 4.5, the SCIP solver not only replicated the structure of the LINGO solution but also achieved better material utilization. SCIP required only **38184.00 cm** of fabric, compared to **38752.32 cm** in the LINGO solution — a saving of **568.32 cm**. Both solutions used two spreadings and achieved full order coverage with 0% excess cutting rate.

In terms of computational performance, SCIP achieved optimality in just **30 search nodes** and **396 iterations**, while LINGO required **3082 iterations**. the difference in solver steps clearly indicates SCIP's superior computational efficiency. Furthermore,

SCIP's solution grouped sizes more compactly in Spreadings 1 and 2, avoiding the fragmented pattern assignment observed in the LINGO layout.

4.4.1.2 Coat Production: SCIP vs. LINGO

The coat production case represents the most complex scenario in this study, involving eight different garment sizes (T48 to T62). The average piece length is 205.49 cm, and the maximum number of fabric plies Kmax is 50. The planning must be executed across four spreadings, all constrained by a 1600 cm table limit.

Table 4.7 shows the LINGO results as reported by Ünal and Yüksel [48], which required over 3 million iterations and achieved a total fabric usage of 85486.51 cm.

Table 4.7: LINGO Result – Coat Case (Ünal & Yüksel)

Sizes	**T48**	**T50**	**T52**	**T54**	**T56**	**T58**	**T60**	**T62**	**No. of piles**	**length, cm**
Total order	37	84	84	84	59	38	22	8		
1. Spreading	0	2	1	2	1	1	0	0	36	1438.48
2. Spreading	2	0	2	0	1	0	2	0	11	1438.48
3. Spreading	0	1	3	1	0	0	0	1	8	1232.98
4. Spreading	3	0	0	0	2	0	0	0	5	1027.48
No. of pieces	37	84	84	84	59	38	22	8	**Total length, cm**	
EQR, %	0	0	0	0	0	0	0	0	**85486.51**	

SCIP was applied to the exact same model and data under a Python-Pyomo environment. The result, displayed in Table 4.8, was a marginally better total fabric usage of 85483.84 cm—saving 2.67 cm compared to LINGO—with a significantly lower computational effort.

Table 4.8: SCIP Result – Coat Case

Sizes	**T48**	**T50**	**T52**	**T54**	**T56**	**T58**	**T60**	**T62**	**No. of piles**	**length, cm**
Total order	37	84	84	84	59	38	22	8		
1. Spreading	0	2	2	2	1	0	0	0	41	1438.43
2. Spreading	2	1	1	1	1	0	0	0	2	1232.94
3. Spreading	0	0	0	0	2	2	0	1	8	1027.45
4. Spreading	3	0	0	0	0	2	2	0	11	1438.43
No. of pieces	37	84	84	84	59	38	22	8	**Total length, cm**	
EQR, %	0	0	0	0	0	0	0	0	**85483.84**	

Although the fabric savings between SCIP and LINGO appear small, the reduction in search effort is dramatic. LINGO's solution took over **3,087,495 iterations**, while SCIP reached optimality with only **2,496 nodes**and **18162 iterations**. This validates

the effectiveness of SCIP in solving large-scale, nonconvex MINLP problems in apparel production.

4.4.1.3 Trouser Production: SCIP vs. LINGO

This case evaluates trousers in five size categories (T24, T26, T28, T30, T32). The average piece length is 112.22 cm, and all constraints remain consistent with the previous models (table length = 1600 cm, 0% ECR). LINGO results from Ünal and Yüksel [48] are shown in Table 4.9, where the solution used two spreadings and achieved a total fabric consumption of 33,667.85 cm in **2549 iterations**.

Note: The maximum number of plies used in this case was $K_{\max} = 100$.

Table 4.9: LINGO Result – Trouser Case (Ünal & Yüksel)

Sizes	**T24**	**T26**	**T28**	**T30**	**T32**	**No. of piles**	**Marker plan length (cm)**
Total Order	40	205	35	10	10		
1. Spreading	2	9	1	0	0	15	1346.71
2. Spreading	1	7	2	1	1	10	1346.71
No. of pieces	40	205	35	10	10	**Total length, cm**	
EQR, %	0	0	0	0	0	**33667.85**	

SCIP, applied to the same dataset, generated a similar result using two spreadings. However, it reduced fabric usage slightly to **33,666.00 cm**, as shown in Table 4.10. While the savings are marginal (1.85 cm), SCIP achieved this in a single node and only and **9 iterations**, showcasing its high performance and efficiency.

Table 4.10: SCIP Result – Trouser Case

Sizes	**T24**	**T26**	**T28**	**T30**	**T32**	**No. of piles**	**Marker plan length (cm)**
Total Order	40	205	35	10	10		
1. Spreading	0	5	1	0	0	35	673.32
2. Spreading	4	3	0	1	1	10	1009.98
No. of pieces	40	205	35	10	10	**Total length, cm**	
EQR, %	0	0	0	0	0	**33666.00**	

Although the difference in material usage is minor, SCIP's computational speed and simplicity of implementation offer strong justification for its adoption in practical applications.

4.4.1.4 Sweatshirt Production: SCIP vs. LINGO

The sweatshirt production problem involves five size categories (S, M, L, XL, XXL), with a maximum table length of 2000 cm and a fabric ply capacity (Kmax) up to 45

layers. Each garment has a uniform average length of 100 cm. The goal is to fulfill all customer demands while minimizing total fabric consumption, without exceeding order tolerances.

Table 4.11: LINGO Result – Sweatshirt Case (Ünal & Yüksel)

Sizes	**S**	**M**	**L**	**XL**	**XXL**	**No. of piles**	**Marker plan length, cm**
Total Order	130	348	444	304	152		
1. Spreading	1	1	6	8	4	38	2000
2. Spreading	3	10	7	0	0	31	2000
No. of pieces	131	348	446	304	152	**Total length, cm**	
EQR, %	0.76	0	0.22	0	0	**138000**	

Table 4.12: SCIP Result – Sweatshirt Case

Sizes	**S**	**M**	**L**	**XL**	**XXL**	**No. of piles**	**Marker plan length, cm**
Total Order	130	348	444	304	152		
1. Spreading	1	1	6	8	4	38	2000
2. Spreading	3	10	7	0	0	31	2000
No. of pieces	131	348	446	304	152	**Total length, cm**	
EQR, %	0.76	0	0.22	0	0	**138000**	

Both SCIP and LINGO generated identical results in terms of total fabric used and the structure of the solution (number of plies, spreading lengths, and piece allocations). However, SCIP achieved this result in just **1609 iterations**, while LINGO required over **57277 iterations** to converge to the same optimum. For both solvers, the Estimated Cutting Ratio (ECR) was 0.76% for size S and 0.22% for size L, confirming the consistency of solution quality across tools.

4.4.1.5 Fabric and solver iterations Comparison: SCIP vs. LINGO

To provide a more intuitive understanding of solver performance, Figures 4.1 and 4.2 present visual comparisons between LINGO and SCIP across the four product categories studied.

Figure 4.1 shows that both solvers achieved very close fabric utilization results across all product categories. Notably, SCIP managed to slightly outperform LINGO in terms of fabric savings for shirts and trousers, with equivalent performance for coats and sweatshirts. These results validate that SCIP can match or even exceed the solution quality of LINGO.

Figure 4.2 highlights a critical advantage of SCIP: computational efficiency. Displayed on a logarithmic scale, the total number of iterations or search nodes used by

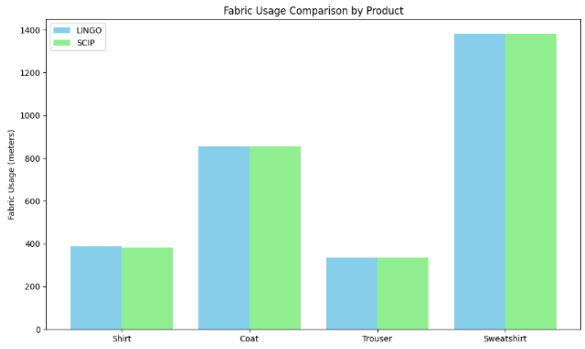


Figure 4.1: Fabric Usage Comparison by Product

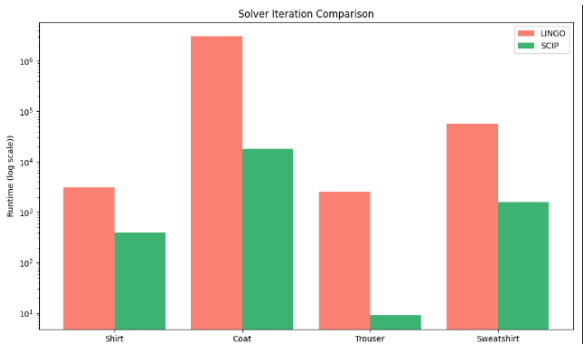


Figure 4.2: Solver iterations Comparison

each solver demonstrates that SCIP required significantly fewer computational steps across all cases.

While LINGO used over three million iterations for the coat and sweatshirt cases, SCIP was able to find optimal or near-optimal solutions using only thousands of nodes. In the trouser case, SCIP reached optimality with a single node. This performance can be attributed to SCIP's use of advanced branch-and-bound and presolving techniques, along with strong cutting planes and constraint propagation.

Why SCIP Performs Better:

- **Integrated MINLP Solving:** SCIP natively handles mixed-integer nonlinear problems using dedicated algorithms, without relying on external solvers.
- **Efficient Branch-and-Bound:** SCIP applies a hybrid of constraint programming and branch-and-cut, enabling it to prune large parts of the search space early.
- **Strong Presolving:** The presolving phase eliminates redundant variables and tightens bounds before branching starts, significantly reducing solve time.
- **Flexibility with Cuts and Heuristics:** SCIP generates domain-specific cuts and employs primal heuristics that accelerate convergence.

These advantages make SCIP an effective and scalable open-source alternative to commercial solvers like LINGO, especially in industrial applications such as Cut Order Planning.

Hardware Consideration:

To provide a fair and consistent performance evaluation, solver efficiency is assessed based on iteration counts rather than wall-clock execution time. This choice

was made because the original study by Ünal and Yüksel [48] did not disclose the hardware used, making direct time-based comparisons unreliable. Our tests were performed on a system equipped with an Intel(R) Core(TM) i5-8365U CPU @ 1.60GHz, 16 GB of RAM, and a 64-bit operating system. By focusing on iteration counts—which are unaffected by hardware speed—we ensure a hardware-independent benchmark of solver efficiency.

4.4.2 Solvers Comparison Summary – All Products

Table 4.13: Solver Performance Comparison Across All Products

Product	Solver	Fabric Used (cm)	Solve Time (s)	Nodes
Shirt	SCIP	38184.00	0.15	30
	Bonmin	38184.00	299.86	6669
	Couenne	38184.00	0.96	700
Coat	SCIP	85483.84	3.24	2,496
	Bonmin	—	>7200	>3,000,000 (no result)
	Couenne	—	>7200	>2,500,000 (no result)
Trouser	SCIP	33666.00	0.13	1
	Bonmin	33666.00	2243.32	54641
	Couenne	33666.00	3.05	3,398
Sweatshirt	SCIP	138000.00	0.21	142
	Bonmin	138000.00	1375.55	86258
	Couenne	138000.00	108.17	114,360

Table 4.13 presents a detailed comparison of solver performance across the four product cases analyzed in this study: shirt, coat, trouser, and sweatshirt. The SCIP solver consistently outperformed Bonmin and Couenne in terms of both computation time and reliability. For the shirt and trouser cases, all solvers reached the same optimal solution, but SCIP achieved it in a fraction of the time. Bonmin, while accurate when it converges, suffered from excessive computation time, particularly for trousers and sweatshirts. Notably, both Bonmin and Couenne failed to solve the coat instance within a reasonable time frame (over 2 hours), whereas SCIP provided a feasible solution in just 3.24 seconds. This highlights SCIP's superior scalability and efficiency for complex mixed-integer nonlinear programming problems in the Cut Order Planning context.

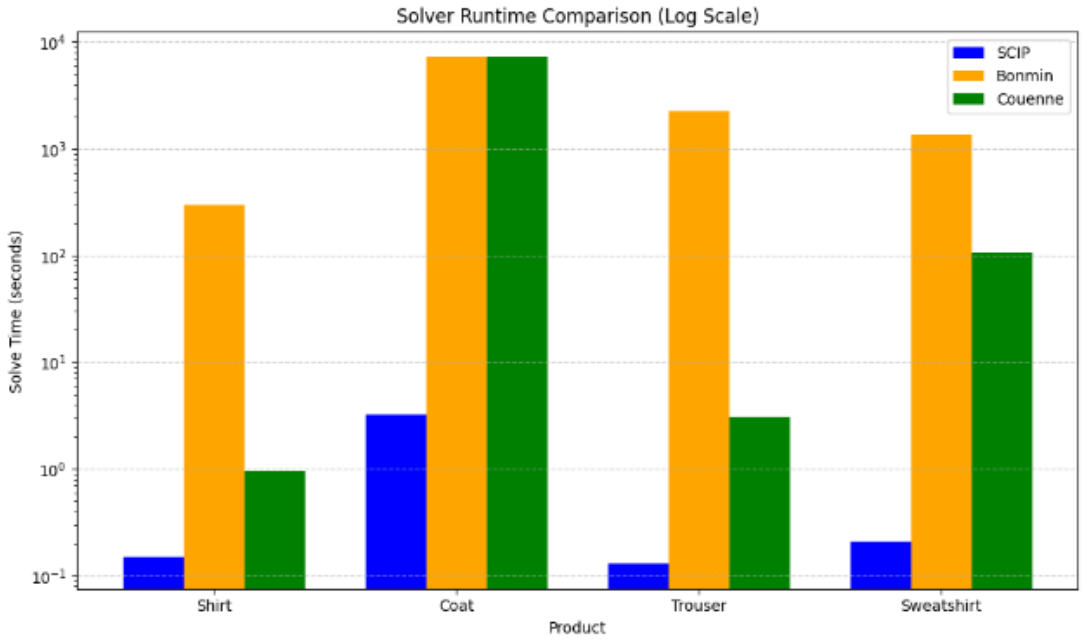


Figure 4.3: Comparison of solver runtime between SCIP, BONMIN and COUENNE.

Comparison of solver runtime between SCIP, BONMIN and COUENNE

Figure 4.3 illustrates the runtime performance of SCIP, Bonmin, and Couenne across the four Cut Order Planning (COP) scenarios. As clearly shown, SCIP consistently outperforms the other solvers in terms of computational speed. For all products, SCIP achieved optimal solutions in less than 4 seconds, while Bonmin and Couenne required significantly more time—often by several orders of magnitude. In the trouser and sweatshirt cases, for example, Bonmin needed over 2243 and 1375 seconds, respectively, whereas SCIP solved both in under a second. Most notably, both Bonmin and Couenne failed to return a result for the coat case even after two hours, highlighting their scalability limitations. This remarkable performance of SCIP is attributed to its efficient branch-and-bound framework, enhanced by aggressive presolving, symmetry detection, and dynamic cut generation. These features enable SCIP to handle large-scale MINLPs effectively, making it the most practical and reliable solver for real-world apparel production planning.

4.5 Conclusion

This chapter provided an in-depth evaluation of various optimization solvers applied to the Cut Order Planning (COP) problem in the apparel industry. Through a series of comparative experiments on four distinct product categories—shirt, coat, trouser,

and sweatshirt—we demonstrated the strengths and limitations of three widely used solvers: SCIP, Bonmin, and Couenne.

The results clearly show that SCIP outperforms both Bonmin and Couenne in terms of runtime efficiency, robustness, and scalability. SCIP was able to deliver optimal solutions with minimal computational effort, even for the most complex instance (coat), where Bonmin and Couenne both failed to converge within acceptable time limits. In contrast, Bonmin and Couenne struggled with larger problem sizes, requiring excessive time and iterations, or failing to return a solution altogether.

This superior performance is largely due to SCIP's hybrid solving approach, which integrates constraint programming, branch-and-bound, presolving, and symmetry handling. Its ability to exploit problem structure and apply dynamic cutting planes allows it to solve MINLPs more effectively than general-purpose nonlinear solvers.

In summary, this chapter validates SCIP as a reliable and efficient solver for complex COP problems. It offers an open-source, high-performance alternative to commercial tools like LINGO, making it a practical choice.

GENERAL CONCLUSION

This thesis explored the application of advanced mathematical optimization techniques to solve the Cut Order Planning (COP) problem in the apparel industry, a critical phase in garment manufacturing that significantly impacts material utilization and production costs. The work was grounded in the MINLP model proposed by Ünal and Yüksel [48], which was re-implemented in Python using the open-source solver `SCIP`. This shift from proprietary software (LINGO) to a transparent and accessible environment enhanced the model's reproducibility, adaptability, and industrial applicability.

Through a comparative study involving real production scenarios—shirts, trousers, coats, and sweatshirts—we evaluated the performance of three solvers: `SCIP`, `Bonmin`, and `Couenne`. SCIP consistently outperformed the others in terms of runtime, scalability, and solution quality. It successfully solved all instances, including the computationally intensive coat scenario where Bonmin and Couenne failed to converge within a reasonable timeframe. This demonstrated SCIP's robustness and efficiency in handling large-scale MINLP problems with both nonlinear and integer constraints.

The literature review highlighted critical research gaps, such as the limited use of open-source solvers, underutilization of full MINLP formulations, and lack of reproducibility in existing studies. This thesis contributes to addressing these gaps by providing a validated, accessible implementation that aligns closely with real industrial data.

The evolution of optimization methods was also thoroughly reviewed, from classical LP and MILP models to nonlinear and mixed-integer nonlinear formulations. Special emphasis was placed on solver strategies such as branch-and-bound, cutting planes, and heuristics like local branching and Relaxation Induced Neighborhood Search (RINS).

The mathematical model was validated against both LINGO benchmarks and practical manufacturing data, confirming its feasibility, logic, and real-world relevance. The shift to open-source tools not only improved computational performance but also democratized access to advanced optimization techniques for small and medium-sized enterprises.

Future work may explore hybrid methods that combine exact and heuristic techniques—such as genetic algorithms, simulated annealing, or local branching—to improve scalability on large datasets. Additionally, the integration of **neural networks** and **machine learning approaches** holds significant potential. These techniques could assist in learning cutting patterns, predicting near-optimal marker configurations, or guiding solver heuristics based on prior problem structures. For instance, deep learning models or graph neural networks (GNNs) could be used to predict promising branching variables or approximate relaxation bounds, helping reduce solver time in complex MINLP scenarios. Incorporating uncertainty modeling through stochastic programming, or developing adaptive systems that respond to real-time production changes, could further enhance robustness and responsiveness in dynamic manufacturing environments.

Overall, this thesis demonstrates that a rigorous, open-source, and optimization-based approach to Cut Order Planning can yield tangible benefits in both operational efficiency and research transparency, setting a solid foundation for future exploration in data-driven textile production.

APPENDIX A
OPTIMIZATION WORKFLOW EXAMPLE

A.1 Introduction

This appendix presents a complete example of how the optimization workflow was implemented using Pyomo and SCIP to solve shirt (COP) model. Each step is explained with accompanying Python code.

A.2 Problem Setup

The input data and problem structure in this example are based on the case study presented by Ünal and Yüksel [48], which addresses the Cut Order Planning (COP) problem in the apparel industry. Their model serves as a reference for the demand distribution and average fabric requirements used here.

We consider an order for four shirt sizes:

Size	Demand
T39	69
T41	96
T43	90
T45	45

Additional Parameters:

- Average fabric per piece: 127.28 cm
- Max spreading length: 1600 cm

- Max plies per spreading: 200
- Number of spreadings: 2

A.3 Step-by-Step Workflow with Code

A.3.1 Step 1 – Model Initialization

```
model = ConcreteModel()
model.I = RangeSet(0, num_spreads - 1)  # spreadings
model.J = RangeSet(0, num_sizes - 1)    # sizes

model.K = Var(model.I, domain=NonNegativeIntegers, bounds=(0, K_max))  #
    plies
model.A = Var(model.I, model.J, domain=NonNegativeIntegers, bounds=(0,
    max_pieces_per_spreading))  # pieces
model.T = Var(model.I, domain=NonNegativeReals)  # marker length
```

Listing A.1: Defining sets and decision variables

A.3.2 Step 2 – Objective Function

```
def objective_rule(model):
    return sum(model.K[i] * model.T[i] for i in model.I)
model.obj = Objective(rule=objective_rule, sense=minimize)
```

Listing A.2: Minimize total fabric used

A.3.3 Step 3 – Constraints

(a) Demand Satisfaction

```
model.demand = ConstraintList()
for j in model.J:
    model.demand.add(sum(model.K[i] * model.A[i, j] for i in model.I) ==
        order[j])
```

(b) Spreading Length Limit

```
model.spread_length = ConstraintList()
for i in model.I:
```

```
    model.spread_length.add(sum(model.A[i, j] * B[j] for j in model.J) <= M
        )
```

(c) Marker Length Definition

```
model.marker_link = ConstraintList()
for i in model.I:
    model.marker_link.add(model.T[i] == sum(model.A[i, j] * B[j] for j in
        model.J))
```

(d) Symmetry Breaking

```
model.symmetry_breaking = ConstraintList()
for i in range(num_spreads - 1):
    model.symmetry_breaking.add(model.K[i] >= model.K[i + 1])
```

A.3.4 Step 4 – Solver Execution

```
solver = SolverFactory('scip', executable='/home/lenovo/scipoptsuite-9.2.2/
    build/bin/scip')
result = solver.solve(model, tee=True)
```

Listing A.3: Solving with SCIP

A.3.5 Step 5 – Output Interpretation

```
if result.solver.status == 'ok' and result.solver.termination_condition ==
    'optimal':
    print(f"Total fabric used: {model.obj():.2f} cm")
    for i in model.I:
        k_val = model.K[i].value
        t_val = model.T[i].value
        if k_val > 0:
            print(f"\nSpreading {i + 1}: {k_val} plies, Length = {t_val:.2f
                } cm")
            for j in model.J:
                a_val = model.A[i, j].value
                if a_val > 0:
                    print(f"  Size {j + 1}: {int(a_val)} pieces per ply
                        Total: {int(k_val * a_val)}")
```

Listing A.4: Displaying the optimal result

A.4 Conclusion

This example demonstrates how the optimization workflow was implemented and solved. The results confirm that:

- All size demands are exactly fulfilled.
- Total fabric usage is minimized.
- The solution uses only two spreadings with smart piece allocation.

A screenshot of the console output can be found in the following figure:

```
=== Shirt COP Solution ===
Total fabric used: 38184.00 cm

Spreading 1: 15.0 plies, Length = 1527.36 cm
  Size 1: 3 pieces per ply → Total: 45
  Size 3: 6 pieces per ply → Total: 90
  Size 4: 3 pieces per ply → Total: 45

Spreading 2: 12.0 plies, Length = 1272.80 cm
  Size 1: 2 pieces per ply → Total: 24
  Size 2: 8 pieces per ply → Total: 96
```

Figure A.1: Console output from the COP model solved with SCIP

BIBLIOGRAPHY

[1] Achterberg, T. (2009). *SCIP: Solving Constraint Integer Programs*. Mathematical Programming Computation, 1(1), 1–41. https://scholar.google.com/scholar?q=Achterberg,+T.+(2009).++SCIP:+Solving+Constraint+Integer+Programs.++\protect\unhbox\voidb@x\bgroup\edef.{Mathematical+Programming+Computation}\let\futurelet\@let@token\let\itshapeMathematical+Programming+Computation\egroup,+1(1),+pp.1%E2%80%9341.&hl=en&as_sdt=0&as_vis=1&oi=scholart

[2] Adarshi, G. (2023). *Nonlinear Programming Optimization*. Medium. https://gaurav-adarshi.medium.com/nonlinear-programming-optimization-df65f0576998

[3] Ali, S. (2017). *Integer Linear Programming*. ResearchGate. Retrieved from https://www.researchgate.net/publication/319449795_Integer_Linear_Programming

[4] Al-Masri, E. (2020). *Complexity Theory 101: Problems Classification*. Towards Data Science, Medium. https://towardsdatascience.com/complexity-theory-101-problems-classification-9793f05e42e1

[5] Avci, S., & Topaloglu, S. (2020). MINLP models for the trim-loss problem with nonlinear cost structures. *Optimization Online*. https://optimization-online.org/wp-content/uploads/2020/10/8078.pdf

[6] Backs, S., Jahnke, H., Lüpke, L., Stücken, M., & Stummer, C. (2020). *Supply Chain Strategies of the Apparel Industry in Research: A Literature Review*. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3558419

[7] Balaji, R. (2006). *Gradient-Based Nonlinear Optimization Methods*. University of Colorado. https://civil.colorado.edu/~balajir/CVEN5393/lectures/chapter-11.pdf

[8] Behera, B. K., & Chakraborty, S. (2016). The marker planning problem in apparel cutting: A literature review. *Journal of Fashion Technology Textile Engineering*, 4(3), 1–6.

[9] Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., Mahajan, A. (2013). *Mixed-integer nonlinear optimization*. Acta Numerica, 22, 1–131. Available at: https://www.researchgate.net/publication/259432162_Mixed-integer_nonlinear_optimization

[10] Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine Learning for Combinatorial Optimization: A Methodological Tour. *European Journal of Operational Research*, 290(2), 405–421. https://www.researchgate.net/publication/328997287_Machine_Learning_for_Combinatorial_Optimization_a_Methodological_Tour_d%27Horizon

[11] Berkey, J. O., & Wang, C. (2022). MILP-based approaches for a bin-packing problem with a fixed-plus-linear charge scheme. *ResearchGate*. https://www.researchgate.net/publication/366266574_MILP-based_approaches_for_a_bin-packing_problem_with_a_fixed-plus-linear_charge_scheme

[12] Bertacco, L., Fischetti, M., & Lodi, A. (2007). *A feasibility pump heuristic for general mixed-integer problems*. Discrete Optimization, 4(1), 63–76. https://www.researchgate.net/publication/222662965_A_feasibility_pump_Heuristic_for_general_mixed-integer_problems

[13] Biegler, L. T., Grossmann, I. E., & Westerlund, T. (2012). A compact MINLP tutorial. *Université Paris-Dauphine MINLP Compact Course*. https://jlinderoth.github.io/papers/Bonami-Kilinc-Linderoth-10.pdf

[14] Bonami, P., Kilinc, M., & Linderoth, J. (2012). *Solving Mixed-Integer Nonlinear Programs by QP-Diving*. Technical Report ANL/MCS-P2071-0312, Argonne National Laboratory. Available at: https://optimization-online.org/2012/03/3409/

[15] Burke, E. K., Kendall, G., & Soubeiga, E. (2004). A hybrid approach for flexible employee scheduling. *Computational Optimization and Applications*, 28(1), 31–50.

[16] BYJU'S. (n.d.). *Linear Programming*. https://byjus.com/maths/linear-programming/

[17] Chang, H., Sahinidis, N. V. (2012). *Optimization Formulations and Computational Studies for Cut Order Planning in Apparel Manufacturing*. Argonne National Laboratory Technical Report ANL/MCS-P3060-1112. https://www.mcs.anl.gov/papers/P3060-1112.pdf

[18] Dantzig, G. B. (1987). *Origins of the Simplex Method*. Technical Report SOL 87-5. Systems Optimization Laboratory, Department of Operations Research, Stanford University. https://scispace.com/pdf/origins-of-the-simplex-method-1tuurpfdul.pdf

[19] FICO. (n.d.). *FICO Xpress Optimization*. Retrieved from https://www.fico.com/en/products/fico-xpress-optimization

[20] Fischetti, M., & Lodi, A. (2011). Heuristics in Mixed Integer Programming. *Wiley Encyclopedia of Operations Research and Management Science*. https://homepages.cwi.nl/~dadush/workshop/discrepancy-ip/papers/heuristics-survey-fischetti-lodi-11.pdf

[21] Ghasemiran Foundation. (2023). *Optimizing decision-making in the apparel supply chain using artificial intelligence (AI): From production to retail*. Retrieved from https://www.ghasemiran.org/upload/upload/1690877639-Optimizing-decision-making-in-the-apparel-supply-chain-using-artifi-From-production-to-retail-(W.pdf)

[22] Gomes, T. M., Santos, H. G., & Souza, M. J. F. (2013). *A Pre-Processing Aware RINS Based MIP Heuristic*. Technical Report, Universidade Federal de Ouro Preto, Brazil. Available at: http://www.decom.ufop.br/haroldo/papers/Gomes2013.pdf

[23] Grossmann, I. E. (2002). *Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques*. Carnegie Mellon University. https://egon.cheme.cmu.edu/Papers/GrossmannOptII.pdf

[24] Grossmann, I. E. (2012). Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3(3), 227–252. https://egon.cheme.cmu.edu/Papers/GrossmannReviewNon.pdf

[25] Gurobi Optimization. (n.d.). *Gurobi Optimizer*. Retrieved from https://www.gurobi.com/

[26] Gurobi Optimization. (n.d.). *Integer Linear Programming – Gurobi*. https://www.gurobi.com/faqs/integer-linear-programming/

[27] HiGHS. (n.d.). *HiGHS - High-performance open-source optimization software*. Retrieved from https://highs.dev/

[28] Huang, L., Chen, X., Huo, W., Wang, J., Zhang, F., Bai, B., Shi, L. (2021). *Branch and Bound in Mixed Integer Linear Programming Problems: A Survey of Techniques and Trends*. arXiv preprint arXiv:2111.06257. https://arxiv.org/abs/2111.06257

[29] IBM. (n.d.). *IBM ILOG CPLEX Optimization Studio*. Retrieved from https://www.ibm.com/products/ilog-cplex-optimization-studio

[30] Jonsson, R. (2015). *Introduction to Mixed Integer Programming*. Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU). Available at: https://www.itk.ntnu.no/_media/emner/fordypning/ttk16/introductiontomip2015.pdf

[31] Karamanov, M. (2006). *Branch and Cut: An Empirical Study*. PhD Dissertation, Carnegie Mellon University. https://www.cmu.edu/tepper/programs/phd/program/assets/dissertations/2006-operations-research-karamanov-dissertation.pdf

[32] Khan, M. I., et al. (2022). *Nonlinear Programming Solvers for Unconstrained and Constrained Optimization Problems: A Benchmark Analysis*. International Journal of Applied Science and Engineering, 19(1), 1–18. https://www.researchgate.net/publication/359890460_Nonlinear_Programming_Solvers_for_Unconstrained_and_Constrained_Optimization_Problems_a_Benchmark_Analysis

[33] Kong, S., Lee, Y., Park, J. (2025). MILP-based line balancing and scheduling in garment production. *Journal of Manufacturing Systems*, In Press. https://www.researchgate.net/publication/388657748_Line_Balancing_in_the_Modern_Garment_Industry

[34] Kunwar, R., & Sapkota, H. P. (2022). An Introduction to Linear Programming Problems with Some Real-Life Applications. *European Journal of Mathematics and Statistics*, 3(2). https://doi.org/10.24018/ejmath.2022.3.2.108

[35] Lal, A. (2021). *Interior-point method for NLP*. Cornell University Optimization Wiki. https://optimization.cbe.cornell.edu/index.php?title=Interior-point_method_for_NLP

[36] Lancia, G., Serafini, P. (2018). *Integer Linear Programming*. In *Compact Extended Linear Programming Models*. Springer. https://www.researchgate.net/publication/319449795_Integer_Linear_Programming

[37] LeanStitch. (2024). *What is a Cut Order Plan?*. Retrieved from https://leanstitch.com/what-is-a-cut-order-plan/

[38] Liyanage, R., Perera, H., & Karunananda, A. (2020). A genetic programming approach to optimize workflows in textile industry. *International Journal of Advanced Manufacturing Technology*, 108(1–2), 399–414.

[39] Lundell, A., Kronqvist, J., & Westerlund, T. (2023). *The Supporting Hyperplane Optimization Toolkit for Convex MINLP*. Optimization and Engineering. Available at: https://optimization-online.org/wp-content/uploads/2018/06/6680.pdf

[40] Orosz, R. P., & Koch, T. (2020). *A Survey on Mixed-Integer Nonlinear Programming Techniques*. arXiv preprint arXiv:2003.09437. https://arxiv.org/pdf/2003.09437

[41] Padamwar, B. V., & Pandey, H. (2019). Optimization Techniques in Operations Research: A Review. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 10(1), 746–752. https://turcomat.org/index.php/turkbilmat/article/view/14604

[42] Paşayev, N. (2010). Investigating the effects of production planning on fabric costs in confection production. *Tekstil ve Konfeksiyon*, 20(3), 262–270. https://www.researchgate.net/publication/289660196_Investigating_the_effects_of_production_planning_on_fabric_costs_in_confection_production

[43] Python Software Foundation. (n.d.). *The Python Programming Language*. Available at: https://www.python.org/ [Accessed June 2025].

[44] Pyomo Documentation. (n.d.). *Pyomo: Python Optimization Modeling Objects*. Available at: https://pyomo.readthedocs.io/en/stable/ [Accessed June 2025].

[45] Rose, D., & Shier, D. R. (2007). Cut scheduling in the apparel industry. *Computers Operations Research*, 34(3), 623–637. https://www.researchgate.net/publication/223628115_Cut_scheduling_in_the_apparel_industry

[46] Singh, M. K. (2020, April 17). *Advantages and Drawbacks of Linear Programming*. Faculty BBA.

[47] Textile Engineering. (n.d.). *Cut Order Planning in Garment Manufacturing*. Retrieved from https://textileengineering.net/cut-order-planning-in-garment-manufacturing/

[48] Ünal, C., & Yüksel, A. D. (2020). *Cut Order Planning Optimisation in the Apparel Industry*. Fibres Textiles in Eastern Europe, 28(1), 8–13. https://doi.org/10.5604/01.3001.0013.5851

[49] Utkün, E. (2016). Model and marker plan effect on productivity in apparel production: A case from bathrobe manufacturing. *Tekstil ve Konfeksiyon*, 26(2), 123–130. https://www.tekstilvemuhendis.org.tr/en/2016_-volume-23-/104/a_study_on_effects_of_model_and_marker_plan_differences_on_fabric_productivity-_case_of_bathrobe---doi-_10-7216-1300759920162310404

[50] Vashishtha, T. (2021). *Complexity Theory 101: Problem Classification*. Towards Data Science. https://towardsdatascience.com/complexity-theory-101-problems-classification-9793f05e42e1

[51] Wascher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3), 1109–1130. https://www.mansci.ovgu.de/mansci_media/publikationen/2007/typology-EGOTEC-5t0pvr6fjifln4r4oav60tt612.pdf