University of Nottingham
UK | CHINA | MALAYSIA

Pattern based learning and optimisation for online packing problems

Thesis

Thesis submitted to the University of Nottingham for the degree of **Doctor of Philosophy, 2024.**

Huayan Zhang

20308921

Supervised by

Ruibin BAI
Jiawei LI
Tieyan LIU

Signature _____

Date _____ / _____ / _____

Abstract

Patterns and their identification are essential tasks in data mining applications. However, few studies have systematically investigated the dynamics of pattern values or their reuse under varying conditions. When problem conditions change, previously effective patterns may become less effective, leading to suboptimal solutions. Therefore, pattern-based optimisation algorithms must not only identify good quality patterns but also adapt to dynamic environments or imperfect predictions through dynamic pattern valuations and adaptive learning for their combinations.

This thesis focuses on pattern-based approaches for solving online COP. Although high-quality patterns can guide the generation of high-quality online solutions, existing research indicates that even minor errors can lead to a decline in the quality of the guided online solution, especially when dealing with dynamic problems. Accordingly, this thesis proposes a two-stage framework, Plan-and-Pack, for online COP, which consists of learning and predicting future distributions during the planning phase to generate high-quality patterns, as well as adaptive decision-making based on observations and online learning during the online decision phase.

This thesis proposes a novel scheme that connects data mining and duality theory in operations research for the efficient identification of patterns and dynamic quantification of their values. This method evaluates patterns based on their ability to meet stochastic constraints and their impact on the objective value. Additionally,

this thesis addresses the updating of patterns to fit new realities through an online adjustment stage, where risky patterns are identified and replaced before packing.

This thesis primarily focuses on packing problems, including the classical one-dimensional online BPP and the more practically relevant Operation Room Scheduling Problem (ORSP). The methods proposed in this thesis have achieved state-of-the-art performance in terms of objective value in both cases. Furthermore, an in-depth analysis of the contributions to performance is also provided.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Ruibin Bai, Jiawei Li, and Tie-yan Liu, for their invaluable guidance, support, and encouragement throughout my PhD journey. Their expertise and insights have been instrumental in shaping my research and enhancing my academic experience.

I am also profoundly thankful to my parents and family for their unwavering support and love, especially during challenging times. Their belief in me has been a constant source of motivation.

I extend my appreciation to my teammates and classmates for their academic help and engaging discussions. The collaborative environment we fostered has enriched my learning experience and contributed significantly to my research.

Lastly, I would like to thank the UNNC Digital Port Lab and the MSRA research scholarship for providing the resources and support that made this research possible.

Contents

Abbreviations

APR Asymptotic Performance Ratio.

APTAS Asymptotic Polynomial-Time Approximation Scheme.

BF Best Fit.

BFD Best-Fit Descending.

BPP Bin Packing Problem.

BW Bounded Waste.

CGAA Column Generation Adaptive Allocation.

CGPP Column Generation Plan-and-Pack.

COP Combinatorial Optimisation Problem.

CR Competitive Ratio.

FF First Fit.

FFD First-Fit Descending.

FPP Fuzzy PatternPack.

KDE Kernel Density Estimation.

LW Linear Waste.

OBPP Online Bin Packing Problem.

ORAPA Online Risk-Aware Pattern Adjustment.

ORL Reinforcement Learning Benchmarks for Online Stochastic Optimisation Problems.

ORSP Operation Room Scheduling Problem.

PP Perfect Pack.

ProP ProfilePack.

PTAS Polynomial-Time Approximation Scheme.

PtnP PatternPack.

RMP Restricted Master Problem.

SMKP Stochastic Multiple Knapsack Problem.

SS Sum-of-Square.

VSBPP Variable Sized Bin Packing Problem.

Chapter 1

Introduction

1.1 Online Packing Problems

The Combinatorial Optimisation Problem (COP) refers to a class of problems closely linked to practical scenarios such as supply chain management, manufacturing, and cloud computing. The objective of such problems is to identify an optimal solution from a finite set of possible choices that maximises the value of a given objective function while satisfying all relevant constraints. As they often involve discrete variables or constraints, COPs frequently exhibit highly nonlinear characteristics, with a solution space that grows rapidly with problem size. Furthermore, high-quality feasible solutions—or optimal ones—are often sparsely distributed within this space, requiring considerable computational effort to locate them efficiently.

Among these, Bin Packing Problem (BPP) is a classic Combinatorial Optimisation Problem, aiming to pack a set of given items into the smallest possible number of bins without exceeding the capacity limitation of each bin. This problem family typically embodies the defining features of online stochastic COPs, as previously outlined, while also presenting well-defined objective functions and

explicit constraint structures. Such structural clarity is widely recognised as facilitating systematic algorithmic development and enabling rigorous performance evaluation.

Traditional research focuses predominantly on offline deterministic formulations, where all problem parameters are known, including objective function coefficients and constraint specifications. The solutions can be obtained through single-step optimisation procedures. These classical formulations may be categorised as offline deterministic COP. However, such deterministic assumptions and offline ones often prove inadequate for real-world applications. First, partial or complete parameter stochasticity may lead to performance degradation or even infeasibility when solutions derived under specific parameter instantiations are applied to structurally similar problems with different parameter realisations. Second, numerous practical COP instances exhibit temporal dependencies, where decision-relevant information is revealed sequentially, requiring global solutions to be constructed step by step. These problems are classified as online stochastic COP.

As a classic COP, the BPP can generally be studied in both online and offline scenarios. However, real-world applications of BPP often prioritise online decision-making. For instance, in cloud computing, this problem can be modelled as a packing problem where the capacity represents the resources (CPU, bandwidth, storage, etc.) offered by service providers, and items correspond to dynamically arriving user requests. In such applications, service providers must allocate resources in real time to accommodate continuously updated user demands. Naturally, the allocation process must be computationally efficient to minimise decision time and ensure responsiveness. Moreover, due to the uncertainty of future demands, the system must maintain robustness across rapidly changing dynamic scenarios. These characteristics introduce significant challenges in online optimisation.

Traditional approaches to solving COP, including solving BPP, have primarily employed two methodological paradigms. The first adopts a rolling-horizon frame-

work, which generates solutions through integer programming solvers or meta-heuristics under assumptions of stationary parameters within finite time windows [Christensen et al., 2019]. These methods frequently require extended computation periods and may exhibit delayed responses to distribution changes. The second paradigm prioritises online efficiency through heuristic algorithms, which are typically employed for rapid decision-making. However, such approaches often suffer from myopic decision strategies that yield suboptimal solution quality. Meanwhile, effective heuristic design usually requires substantial domain expertise. The development of algorithms capable of producing high-quality solutions with rapid decision-making capabilities is recognised as having both theoretical significance and practical value for real-world applications.

This thesis focuses on Online Bin Packing Problem (OBPP), where items arrive sequentially without prior knowledge, requiring immediate packing decisions while respecting bin capacity constraints. Although most existing research adopts zero-knowledge heuristics that disregard future item distributions, real-world applications often allow decision-makers to leverage historical data and domain knowledge to estimate future demands, offering potential efficiency gains over conservative zero-knowledge approaches. Crucially, if predictions perfectly match reality, the problem reduces to an offline setting, which generally yields superior solutions; however, in practice, predictions are inherently imperfect, raising the key challenge of designing robust online algorithms that effectively exploit probabilistic estimates while mitigating errors. This thesis investigates how to systematically integrate such predictive information into online bin packing algorithms to achieve better performance without sacrificing adaptability to dynamic and uncertain environments.

The classical packing problem comprises two fundamental components: bins and items. Items are abstract elements characterised by volumetric attributes, while bins are containers with finite capacities. In online packing formulations, items

are revealed sequentially over discrete time steps, requiring immediate packing decisions at each stage. The inherent uncertainty stems from the incomplete knowledge of future item sequences, which remains independent of the decision-making process (i.e., packing operations do not influence subsequent item distributions). In particular, retrospective analysis of packed items is permitted, enabling potential utilisation of online learning mechanisms for distribution prediction. Practical implementations may derive prior knowledge of item distributions from historical records through statistical aggregation.

Regarding temporal scope, although theoretical studies frequently consider infinite decision horizons, this research focuses on sufficiently long yet finite sequences to enhance practical relevance. In port logistics operations, for example, containers must be transported promptly to avoid service charges, rendering long-term order considerations operationally insignificant. Furthermore, the analysis is restricted to finite discrete item types, a configuration that facilitates a rigorous solution quality assessment without loss of generality. Continuous item attributes can be effectively transformed into discrete types through appropriate discretisation techniques. The attributes of OBPP are summarised as follows:

1. **External Stochasticity**. Stochastic components exhibit conditional independence from both decision variables and system states.

2. **Weak Distributional Knowledge**. Exact probability distributions may not be available. However, they can be estimated through statistical or machine learning approaches. Such estimates inevitably contain approximation errors relative to true distributions.

3. **Discrete Finite Horizon**. Decision-making occurs through sequential time steps within bounded temporal windows, requiring immediate action determination upon each arrival.

4. **Discrete Finite Types**. System arrivals are constrained to finite discrete

categories. Continuous arrival patterns are accommodated through validated discretisation procedures.

1.2 Applying Patterns for Online Packing Problems

When estimates of future item demand are available, a natural approach is to treat the scenario as an offline problem and solve it accordingly, with the algorithm subsequently attempting to reconstruct the precomputed offline solution during online decision-making—an effective methodology when estimates are accurate and solutions exhibit high quality. However, this approach becomes problematic as the prediction horizon extends, due to the inherent difficulty in maintaining forecast precision. Reliance on estimated solutions may disrupt online operations and degrade system-wide performance, especially since even perfect demand estimation cannot compensate for poor solution quality. Addressing these limitations, this thesis advocates a pattern-based approach, where reusable high-performance patterns derived from demand forecasts guide online decisions through three principal mechanisms: their combinatorial potential enables robust offline solution construction, their inherent constraint satisfaction (e.g., bin capacity) simplifies decision-making to demand allocation alone, and their dual interpretability-adaptability characteristics permit both dynamic response to environmental fluctuations and cross-configuration applicability.

Patterns represent one of the most potent and effective problem-solving methodologies in computer vision [Lin et al., 2024] and time-series data analysis [Breitenbach et al., 2023]. Pattern-related mining tasks have strong ties to the machine learning community, where the typical research paradigm involves extracting key information (i.e., patterns) by mining large datasets. However, in the context of

COP, this paradigm faces challenges due to the scarcity of high-quality labelled data. Solving COP problems often requires significant computational effort to reach optimal solutions, making it difficult to obtain sufficient labelled data for effective supervised learning-based pattern mining approaches. Conversely, in COP, insights derived from mathematical structural analysis can yield valuable metrics that guide the pattern mining process. Shadow prices constitute a commonly used metric in this regard; in linear programming, they quantify the change in the objective function value resulting from minor adjustments to constraints. In the case of OBPP, item demands implicitly constrain the frequency with which different patterns are reused. Thus, shadow prices can measure the influence of item demands on the objective function, providing a robust basis for identifying high-quality patterns. Based on this observation, this thesis proposes utilising shadow prices to evaluate the value of patterns and employs this metric to conduct pattern mining and construct packing plans that guide online bin packing.

The inherent randomness and uncertainty of online problems necessitate dynamic adjustments in decision-making to mitigate these effects and achieve efficient solutions. Due to unpredictability, the initial packing plan may not be entirely reliable, necessitating ongoing adjustments based on continuous observations of item demands. The validity of a packing plan is determined by its divergence from newly observed item distributions. When deemed unreliable, alternative strategies are essential for packing subsequent items. Two adjustment strategies are developed: firstly, each time a packing decision is made, the reliability of the plan is evaluated. If unreliable, the process switches to zero-knowledge online heuristics as a performance safeguard, known as the on-packing uncertainty handling strategy. Secondly, as observations update, patterns in the existing plan are evaluated using shadow price metrics, allowing preemptive identification and modification of potentially detrimental patterns. This strategy permits adjustments before item arrival, avoiding the myopic nature of zero-knowledge approaches, and is termed the before-packing uncertainty handling strategy. These strategies enable dynamic

adaptation in online bin packing, effectively managing uncertainty with predictive adjustments and real-time decision-making in dynamic environments.

The OBPP finds widespread applications across various fields. Currently, algorithms at the application level are predominantly governed by traditional zero-knowledge heuristics. The feasible solutions generated by these algorithms are often suboptimal, with a significant gap from the optimal solution. Optimising this gap is one of the key motivations for implementing pattern-based packing. This paper will discuss a class of problems closely related to the BPP, namely the Operation Room Scheduling Problem (ORSP), along with corresponding pattern mining and solution construction methods. This type of problem is widely applied in hospital scheduling, and optimising it can significantly reduce operational costs and improve response times for surgeries.

Finally, a review of the literature reveals a deficiency in current academic research regarding the identification and application of patterns in Combinatorial Optimisation Problem (COP). Addressing this gap serves as a motivation for our study. A search on Google Scholar using the keyword "combinatorial optimisation problem" generates approximately 1,240,000 results. However, when searching with the keywords "pattern" and "combinatorial optimisation problem," only about 347,000 results are obtained. Research related to patterns thus comprises just 27% of the total, and even within this subset, a significant amount of content is only tangentially related to solving problems with patterns. A search using more specific keywords, "pattern mining" and "combinatorial optimisation problem," yields only 17,400 results. Only in recent years has the gap in this area attracted attention from researchers. A similar search focusing on studies from the past five years shows that research concerning patterns in combinatorial optimisation problems has increased to about 39% (9,130 of 23,000), with a significant proportion (7,850 results) involving pattern mining methods. Nevertheless, this indicates that solving COPs based on patterns is still not the mainstream approach.

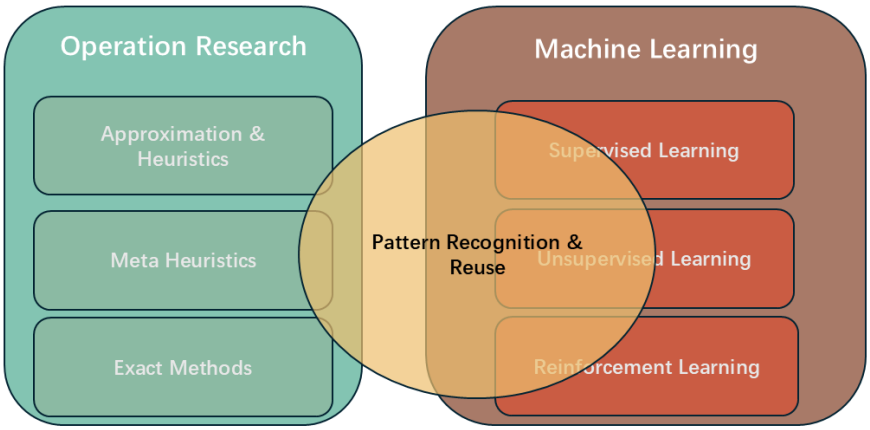


Figure 1.1: Venn graph of research methods in operation research and machine learning community

1.3 Research Questions

This thesis focuses on applying pattern-based online optimisation methodology to address the Online Bin Packing Problem (OBPP). As previously discussed, OBPP is a subset of Combinatorial Optimisation Problems (COPs), characterised by challenges such as high-dimensional solution spaces and NP-hardness. Hence, efficient pattern mining methods must both identify effective patterns and operate within feasible time limits. This leads to our first research question (RQ1), which simultaneously addresses the quality of patterns and the efficiency of the mining process.

The second core challenge arises from the inherent stochasticity of online problems, where predictions often contain errors. This implies that, given a high-quality pattern-based packing plan, we must not only follow and restructure efficient packing plans but also make adaptive online decisions based on continually updated observations. These aspects will be addressed by research question 2 (RQ2) and research question 3 (RQ3).

Finally, this work attempts to extend its application to more complex situations, including dynamic item distributions and more intricate real-world settings. This poses research question 4 (RQ4), which focuses on the generalisability and transferability of the proposed pattern-based online optimisation method.

The detailed research questions are listed as follows:

1. **How to Recognise Patterns?** The application of pattern-based approaches to OBPP requires formal pattern definition and high-quality pattern identification. This primary research question decomposes into three sub-problems:

   (a) **Definition of Pattern in OBPP.** A systematic pattern representation must be designed to align with OBPP's structural properties, facilitating both analytical investigations and algorithmic implementations.

   (b) **Identifying High-Quality Patterns.** High-quality solutions are hypothesised to emerge from optimised pattern combinations, requiring evaluation metrics that account for extended decision horizons to prevent myopic optimisation.

   (c) **Efficient Pattern Mining.** Given the exponential search spaces inherent to COP, developing computationally tractable pattern discovery methods presents a fundamental challenge.

2. **How to Guide Online Decisions with Patterns?** Given a predefined pattern set, online packing decisions must determine both item-bin assignments and pattern replication strategies, necessitating integrated decision frameworks that balance immediate actions with long-term packing efficiency.

3. **How to Handle Uncertainty for OBPP?** The inherent stochasticity of OBPP induces prediction inaccuracies that might yield suboptimal pat-

terns, potentially compromising solution quality. This necessitates robust uncertainty mitigation mechanisms:

(a) **Handling Uncertainty On-Packing.** Real-time decision-making must dynamically adapt to prediction errors and distributional shifts during active packing operations.

(b) **Handling Uncertainty Before Packing.** Continuously updated information streams should be leveraged to proactively identify and reconfigure low-quality patterns before the packing decision.

4. **Can the Pattern-Based Method Apply to Real-World Applications?** Practical implementation remains critical for COP research. This thesis examines two application paradigms:

(a) **OBPP with Real-World Distributions.** Incorporating complex distributional patterns observed in operational environments, including periodic and transient demand fluctuations.

(b) **Operation Room Scheduling Problem (ORSP).** Addressing operating room scheduling complexities through enhanced uncertainty modelling of surgical durations and resource constraints.

1.4 Research Outcome and Contributions

The core contribution of this thesis lies in the application of the concept of patterns, which is widely used in other fields, to enhance the solving performance of the half-knowledge online packing problem. On the methodological front, this study treats patterns as constituent elements of solutions and introduces them into the resolution of the COP. This approach generates a set of patterns by mining incomplete information and reusing it to construct a blueprint for online decision-making. Unlike common pattern recognition tasks such as classification

and regression, the inherent uncertainty present in OBPP implies that decisions based on incomplete information may mislead online algorithms. Therefore, it is essential not only to identify high-quality patterns but also to dynamically adjust decisions when predicted information deviates from the actual sequence. This will expand the existing solution paradigms for the relatively underexplored but increasingly recognised area of patterns and their mining techniques in COP.

Regarding the problem itself, this thesis places a greater emphasis on the more practically relevant half-knowledge setting for OBPP. Specifically, it assumes that the distribution of item types based on online learning can predict future frequencies to some extent, despite potential errors. This assumption does not alter the essence of the online packing problem, which stipulates that items must be packed upon arrival. In contrast, other half-knowledge configurations - such as obtaining precise predictions of future items or using buffers to defer packing - would relax the constraints of online decision-making. However, despite being intuitive, prediction-based half-knowledge remains relatively scarce until recent years [Angelopoulos et al., 2023, Lin et al., 2024]. Moreover, these methods often fail to outperform the classical zero-knowledge method, BF, under simpler problem settings. The experiments presented in this thesis demonstrate that, with high-quality patterns and a reasonable handling of uncertainty, this barrier can be overcome, resulting in relatively high-quality solutions. This provides experimental assurance for the performance of half-knowledge methods and instils confidence for further research in this area.

The technical chapters' contributions and their alignment with the research questions are now elaborated. Chapter 4 introduces the plan-and-pack framework, which leverages a distributionally shifted demand estimate to pre-compute packing strategies for online execution. This chapter formalises foundational concepts central to the thesis, including rigorous definitions of patterns and an integer programming model for generating packing plans. While detailed algorithmic

solutions are deferred to subsequent chapters, this work establishes the theoretical groundwork for addressing RQ1 and RQ2. Further, the chapter analyses the plan-and-pack framework's performance under prediction errors, directly engaging with RQ3's focus on pattern-based decision-making under uncertainty. A preliminary experiment demonstrates the feasibility of identifying efficient and robust patterns through error-tolerant sampling sequences. Additionally, the systemic impacts of underestimation and overestimation on number of bins are rigorously characterised, providing analytical bounds on the framework's robustness.

Chapter 5 introduces a novel methodology, termed CGPP, to the field of learning-based online optimisation. By leveraging the mathematical rigour of duality theory and shadow prices from operations research, the proposed method identifies high-quality reusable patterns while precisely quantifying their utility under known uncertainty distributions. This yields near-optimal solutions and achieves significant performance improvements over state-of-the-art online bin-packing algorithms, thereby addressing the core concern of pattern generation outlined in RQ1b. Additionally, the iterative incremental pattern generation approach avoids searching the whole possible pattern space. This helps reduce computational costs and tackles the concern in RQ1c.

During the online packing phase, CGPP incorporates an adaptive threshold-driven fallback and replanning mechanism to mitigate uncertainties arising from demand prediction inaccuracies. This algorithm demonstrates excellent performance in minimising the number of bins, indicating enhanced resilience to forecasting errors. This directly contributes to RQ2 of designing online packing strategies and RQ3a's focus on managing uncertainty during the packing process.

The methodology is further extended to handle instances with unknown uncertainty distributions through an adaptive online learning scheme. By integrating improved distribution forecasting capabilities with robust packing strategies for imperfect plans, the enhanced algorithm achieves notable performance gains over

conventional methods. These advancements substantiate its efficacy in addressing RQ4's emphasis on dynamically evolving distributions, as the framework autonomously refines pattern generation through continuous adaptation.

In situations where there is a considerable bias between predicted demand and actual demand, plans based on these predictions can generate risky patterns. These risky patterns tend to be suboptimal under the actual item sequence, often resulting in incomplete realisation with certain items being overestimated. Although the adaptive strategy designed by CGPP has achieved significant results, its adaptive fallback strategy employs a knowledge-free algorithm that must be executed during the packing stage. This often leads to a delayed recognition of prediction errors and non-optimal patterns. It may also potentially result in the underutilisation of effective patterns. These analyses led to RQ3b, which considers whether it is possible to adjust patterns based on observations before making specific packing decisions, thereby adapting them to newly observed circumstances.

As a reaction to RQ3b, Chapter 6 presents the pattern-and-adjust-and-pack framework. Specifically, an adjustment layer is introduced between the planning and packing decision levels, which dynamically modifies the remnant pattern for half-packed bins according to the discrepancies between the plan stage and the latest predictions. This layer assesses the risk of remnant patterns by evaluating the differences between online pricing and the pricing used at the planning stage, subsequently updating them. This strategy of online analysis and correction of pattern risk is referred to as Online Risk-Aware Pattern Adjustment (ORAPA). Chapter 6 first develops an online pricing mechanism that evaluates the significance of patterns, drawing inspiration from the concept of shadow pricing. This method can be implemented over a shorter duration than rolling-horizon planning and can evaluate open bins of arbitrary sizes. Although the online pricing mechanism is designed for the adjustment of remnant patterns, it also aids in identifying good and bad patterns within a dynamic environment, as specified in RQ1b.

In Chapter 6, the generation of new remnant patterns is modelled as Stochastic Multiple Knapsack Problem (SMKP). This modelling takes into account the inherent variant-size bin characteristics of remnant patterns. Two methods have been developed to solve SMKP: Fast-GA and the Gurobi integer programming solver. The computational costs and solving performance of both methods are also evaluated. The results indicate that ORAPA helps exceed comparative baselines under various complex distributions and unknown actual distributions, particularly in relation to the performance of CGPP without the adjustment layer.

In terms of balancing computational cost and solving performance, Fast-GA demonstrates a more balanced performance, while the Gurobi solver often achieves superior objective solutions at the expense of higher time costs. These conclusions address the concerns of RQ2b, making it possible to intervene and adjust patterns in advance based on certain observations. Regarding the dynamic distributions of interest in RQ3, the strategy of adding an adjustment layer also enhances responsiveness to these dynamic distributions. Finally, the reduced runtime ensures that the time costs associated with pattern generation, as highlighted in RQ1, remain within a manageable range.

In Chapter 5, preliminary discussions on the extension of RQ4b to more practical issues are conducted. In Chapter 7, the methods used to solve OBPP are adapted to the more practical ORSP. The main characteristic of ORSP is the inclusion of various uncertainties, which encompass both quantity uncertainties arising from whether surgeries are performed and time uncertainties pertaining to the surgeries themselves.

To transfer the aforementioned methods to ORSP, the duration of surgeries is discretised using time intervals, thereby converting the multivariate uncertainties into uncertainties related to the number of time intervals. This strategy transforms ORSP into an approximate OBPP problem, enabling the application of existing methods. A variant algorithm, CGAA, is proposed to address this transformed

problem, achieving results superior to state-of-the-art performance on both real hospital data and data synthesised from real data. This outcome also demonstrates that, under appropriate transformations, effective methods for OBPP can be adapted to applications such as scheduling.

The work presented in this thesis has led to three publications, including two journal papers and one conference paper: Chapter 4 and 5 were developed into the journal paper 'Pattern-based learning and optimisation through pricing', submitted to *Pattern Recognition* and is currently under major revision ([Zhang et al., 2024]). Chapter 6 was expanded into the journal paper 'Online Risk-Aware Pattern Adjustment for Bin Packing Problem', submitted to *Expert Systems with Applications*. Chapter 7 was condensed into the conference paper 'A Two-Stage Plan-and-Allocate Algorithm for Operating Room Scheduling Problem with Uncertainties', which was published at the IEEE WCCI 2024 conference.

1.5 Thesis Outline

The detailed outline and research content of the thesis are illustrated in Fig 1.2. This document is primarily divided into two sections: the theoretical analysis (Chapter 2) and literature review (Chapter 3) of the studied OBPP problem, and the specific methodological contributions. In the theoretical analysis and literature review section, the mathematical model of the classic deterministic offline BPP and its related approximation analyses of both offline and online algorithms will be presented. As the study of OBPP is part of the broader research on COP, the literature review will extend the focus to various problem variants and solution methods within the entire COP research landscape.

This research primarily comprises four main components: the solution framework, pattern generation, online packing, and the target problem. In the framework section, this thesis presents the overall plan-and-pack framework and further in-

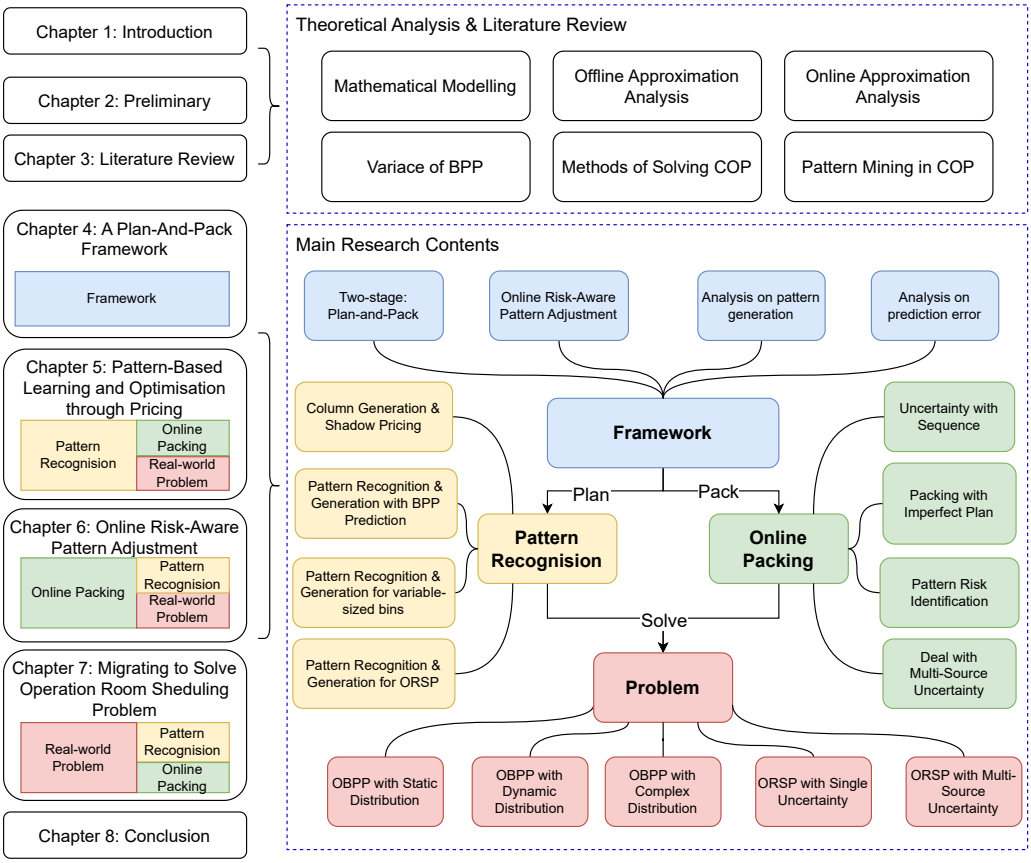


Figure 1.2: Research Content and Scope

troduces a variant that incorporates online risk-aware pattern adjustment. This thesis also analyses the feasibility of obtaining optimal online solutions through pattern generation and the potential impact of erroneous predictions on the framework.

Pattern generation and online packing are two critical components of the algorithmic framework**,** whose implementations vary across different problems and application contexts. Consequently, discussions and analyses of these two aspects are integrated throughout all technical chapters. The study of pattern generation focuses on efficient pattern identification based on pricing and pattern recognition under dynamic distributions. Additionally, this thesis addresses variable-sized bin pattern generation for replacing risky patterns. Lastly, this content involves pattern recognition and generation for Operation Room Scheduling Problem, a

more complex real-world problem. The online packing research primarily focuses on managing prediction errors. This includes identifying prediction errors and ensuring algorithm performance under imperfect plans. This section also covers identifying risky patterns based on the latest predictions. Given Operation Room Scheduling Problem's more complex uncertainty structure, this thesis incorporates algorithms for multi-source uncertainty into the discussion on online packing.

This thesis mainly examines various variants of online packing problems, categorised primarily by their distributions. These distributions include both static and dynamic types, with static distributions comprising classical discrete distributions such as uniform and discretised normal distributions, as well as more complex distributions frequently encountered in practical problems, such as dual normal and Weibull distributions. Dynamic distributions involve mixtures of different distributions over time scales. It also considers variants of Operation Room Scheduling Problem that arise from different sources of uncertainty.

Chapters 4-7 constitute the core technical chapters of this thesis, presenting the proposed methodologies, experimental results, and associated analyses. Chapter 4 establishes the foundation of the plan-and-pack framework and validates its operational feasibility. Chapter 5 presents a systematic approach to pattern identification under partial observability through shadow pricing and column generation. Building upon this foundation, Chapter 6 focuses on enhancing online decision-making performance through adaptive pattern refinement, introducing dynamic adjustment strategies that iteratively improve the overall framework's robustness to distribution shifts. Finally, Chapter 7 addresses the Operation Room Scheduling Problem by developing a transformation methodology that converts multi-source uncertainties inherent in surgical scheduling into a pattern-based planning paradigm analogous to OBPP, thereby extending the framework's applicability to complex real-world scenarios.

Chapter 2

Preliminary

2.1 Classifying Bin Packing Problem

Traditionally, the Bin Packing Problem (BPP) is classified into offline and online problems based on the decision-making context [Coffman et al., 2013]. This classification depends on whether bin-packing decisions need to be made online. However, as pointed out in Chapter 1, many real-world scenarios require the ability to make online decisions with some knowledge of the item sequence. This necessitates an expansion of the traditional online-offline dichotomy. Based on the availability of sequence information to the decision-maker, this thesis introduces a three-category classification for BPP: zero-knowledge, full-knowledge, and half-knowledge configurations. In the zero-knowledge setting, corresponding to online problems, algorithms do not utilise item sequence information. The full-knowledge setting corresponds to offline problems, where algorithms have complete information about the sequence to be packed. Finally, the half-knowledge configuration, which forms the focus of this thesis, maintains certain online decision capabilities while using partial item sequence information or predictions to enhance online decision performance.

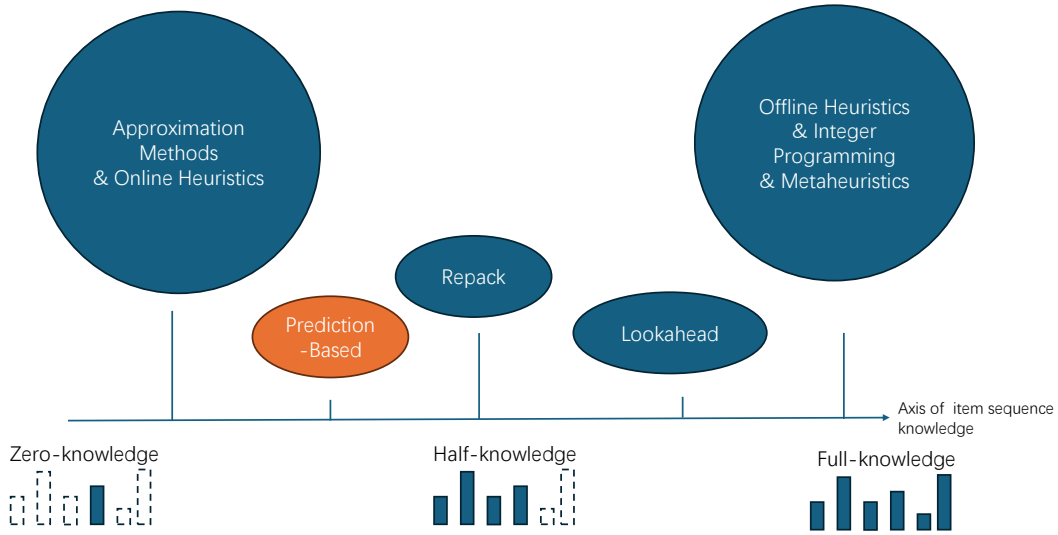


Figure 2.1: Classification of research on packing problems based on information accessibility

Full-knowledge configuration assumes complete prior knowledge of item sequences, enabling globally optimal solutions through offline optimisation. The primary challenge lies in combinatorial explosion within extended sequences, which is addressed through hybrid approaches combining commercial solvers (e.g., Gurobi, CPLEX) with metaheuristic and hyperheuristic methodologies [Glover and Sorensen, 2015]. While theoretically optimal, the impracticality of perfect information assumptions severely limits real world applicability.

Existing literature has predominantly focussed on the two extreme configurations, leaving half-knowledge configurations substantially underexplored. Half-knowledge configurations relax zero-knowledge constraints while maintaining partial information restrictions. Such relaxations include limited repacking [Galambos and Woeginger, 1995] or partial future information access [Grove, 1995]. Our work emphasises a more natural paradigm in which historical partial sequences inform predictive models for enhanced decision-making under strict online constraints (no repacking, unknown future sequences). These approaches are called prediction-based methods. Existing research demonstrates that accurate predic-

tions can improve online performance [Antoniadis et al., 2020].

The scarcity of half-knowledge research becomes evident when examining Coffman et al. [2013]'s survey of 185 studies, which includes only 28 addressing half-knowledge configuration. Within OBPP specifically, prediction-based methodologies have gained traction only recently [Angelopoulos et al., 2023, Lin et al., 2022]. This research gap motivates the investigation in this thesis, particularly regarding the integration of machine learning with combinatorial optimisation under partial observability constraints.

The following sections present the terminology, metrics, and important theoretical results related to the Bin Packing Problem. As previously mentioned, extensive theoretical work is based on the full-knowledge (offline) and zero-knowledge (online) problem settings, with research on half-knowledge partly building upon these outcomes. Thus, Sections 2.2 to 2.3 introduce these aspects to provide readers with the necessary background knowledge. Subsequently, Section 2.4 presents some background and discussion on the half-knowledge configuration.

2.2 Offline Bin Packing Problem

2.2.1 Formulation

Bin Packing Problem (BPP) is one of the most studied COP, which is highly associated with task scheduling and resource allocation[Coffman et al., 2013]. There are two critical elements in such a problem: bins and items. A bin is a container with a constant capacity that allows items to be allocated into it, while an item has a feature of size. Given a non-empty series of items, BPP aims to use as few bins as possible to let all items be packed into bins while the capacity of bins is not exceeded.

A standard formalisation of the Bin Packing Problem (BPP) is presented for further discussion. Consider a sequence of items $L = (i_1, i_2, ...)$ with N items, where the size of items is denoted by s_i. The capacity of the bins is a constant B. A commonly adopted practice is normalising the capacity to 1 and limiting the sizes within the range [0, 1). Let the decision variable y_j represent whether bin j is utilised in a solution ($y_j = 1$) or not ($y_j = 0$). For simplicity, let M denote the maximum number of bins allowed. The decision variable x_{ij} signifies whether the i-th item will be packed into bin j. The classical 1D BPP formulation [Martello, 1990] is described by Eq. (2.1)-(2.5).

$$\min \sum_{j=1}^{M} y_j \tag{2.1}$$

$$s.t.$$

$$\sum_{i=1}^{N} s_i x_{ij} \leq B y_j \qquad \forall j \in \{1..M\} \tag{2.2}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad \forall i \in \{1..N\} \tag{2.3}$$

$$y_j \in \{0, 1\} \qquad \forall j \in \{1..M\} \tag{2.4}$$

$$x_{ij} \in \{0, 1\} \qquad \forall j \in \{1..M\}, \forall i \in \{1..N\} \tag{2.5}$$

Decision variables x and y are 0-1 integer variables as Eq. (2.4)-(2.5) represent. The objective (2.1) represents the goal of BPP to minimise the number of bins used. Constraint (2.2) states that the allocation of items should not exceed the capacity of bins. Constraint (2.3) ensures that items will be packed exactly once across all bins. For a single bin, the *level* represents how much space is filled with items of a bin, while the *waste* represents the unfilled part. Formally, they are defined as follows:

Definition 2.1. *Level of bin j.* $\mathtt{L}(j) = \sum_i^N s_i x_{ij}$

Definition 2.2. *Waste of bin j.* $\mathtt{W}(j) = B - \mathtt{L}(j) = B - \sum_i^N s_i x_{ij}$

2.2.2 Performance Metrics

The Bin Packing Problem is NP-hard[Garey and Johnson, 1979], meaning it cannot be solved in deterministic polynomial time. Conversely, practical applications of the BPP, such as memory pagination and file allocation, require sufficiently good solutions within limited computational constraints. This has driven the investigation of polynomial-time approximation algorithms. For offline problems, performance is assessed using the worst-case performance ratio. Let $\mathcal{A}(L)$ denote the number of bins used by algorithm $\mathcal{A}$ on the item sequence L, and $\mathcal{OPT}$ represent the optimal number of bins for the item sequence L. Let V_α be the set of all lists with a maximum item size bounded by a positive $\alpha \leq B$, the *absolute ratio* is defined as:

Definition 2.3 (Absolute Ratio[Coffman et al., 2013])**.**

$$R_{\mathcal{A}}(\alpha) = \sup_{L \in V_\alpha} \left\{ \frac{\mathcal{A}(L)}{\mathcal{OPT}(L)} \right\} \tag{2.6}$$

For problems with sufficiently large N, the *asymptotic worst-case ratio* or *asymptotic performance ratio*(APR) is used to analyse the algorithm's performance. This metric considers the limit of performance when the length of the item sequence approaches infinity. Let $k \geq 1$ be the objective value of the optimal algorithm, the *asymptotic worst-case ratio* is defined as:

Definition 2.4 (Asymptotic Performance Ratio (APR)[Coffman et al., 2013])**.**

$$R^{\infty}_{\mathcal{A}}(\alpha) = \lim_{N \to \infty} \sup_{L \in V_\alpha} \left\{ \frac{\mathcal{A}(L)}{\mathcal{OPT}(L)} \right\} \tag{2.7}$$

Generally, better algorithms have a smaller ratio. It is also clear that the ratio has a lower bound of 1 and $R_{\mathcal{OPT}} = 1$.

The above definitions also provide tools to determine whether a fast approximation

algorithm exists for the NP-hard problem. The definition of Polynomial-Time Approximation Scheme and Asymptotic Polynomial-Time Approximation Scheme are proposed by Christensen et al. [2017] to describe the approximability of NP-hard problems:

Definition 2.5 (Polynomial-Time Approximation Scheme (PTAS)). *Given a constant approximate factor ϵ, a problem admints a Polynomial-Time Approximation Scheme if for every constant $\epsilon > 0$, there is a poly(n)-time algorithm with $1 + \epsilon$ absolute worst-case ratio where n is the size of the input.*

Definition 2.6 (Asymptotic Polynomial-Time Approximation Scheme (APTAS)). *Given an approximate factor ϵ, a problem is to admit a Polynomial-Time Approximation Scheme if, for every constant $\epsilon > 0$, there is a poly(n)-time algorithm with CR of $1 + \epsilon$ where n is the size of the input.*

It should be noted that, although they belong to the NP-hard category, different problems can exhibit varying degrees of approximability. For example, the classic BPP allows for an APTAS, whereas the travelling salesman problem does not[Christensen et al., 2017], unless the distances are Euclidean[Arora, 1998].

2.2.3 Theoretical Results

Johnson [1973] was the first to systematically investigate a set of approximation algorithms for solving the BPP. These algorithms generate solutions by sequentially packing items based on the item sequence. In each iteration, an item is designated as the *current* item, which is then packed into a non-empty bin following certain rules. If no non-empty bin satisfies the rule, a new empty bin is opened to accommodate the current item. Two renowned rules are represented as follows:

Definition 2.7 (First Fit (FF)). *Pack the current item into the non-empty bin with the lowest index.*

Definition 2.8 (Best Fit (BF)). *Pack the current item into the bin with the highest level that allows the current item to be packed.*

Johnson [1973] proved First Fit and Best Fit has the same APR. The most recent theoretical worst-case performance was made by Dósa and Sgall [2013, 2014]:

Theorem 2.1 (Dósa and Sgall [2013, 2014]). $R^{\infty}_{\mathcal{FF}}(\alpha) = R^{\infty}_{\mathcal{BF}}(\alpha) = 1.7$.

The standard BPP has access to all items. To distinguish from the online version that is mainly discussed in the thesis, it is emphasised that the standard problem configuration to be *offline* BPP. The approximate algorithms that effectively solve the offline BPP usually utilise the full information of the items. Two typical algorithm families are introduced, and the associated theoretical analysis represents how the complete information of items can help develop efficient algorithms.

When items are ordered in ascending order by size, the above online algorithms will achieve worst-case performance. A natural improvement is to permute the items in better order before constructing the solution. Johnson et al. [2006] analysed First-Fit Descending (FFD) and Best-Fit Descending (BFD), where the items are sorted descending by the size and then packed with First Fit and Best Fit rules, respectively. Johnson et al. [2006] proved the APR of the two algorithms to be:

Theorem 2.2 (Johnson et al. [2006]). $R^{\infty}_{\mathcal{FFD}}(\alpha) = R^{\infty}_{\mathcal{BFD}}(\alpha) = \frac{11}{9} \approx 1.22$.

Another approach attempts to solve an approximated problem of the integer programming of BPP defined by Eq.(2.1)-(2.5) to avoid solving the NP-hard integer programming. Let $t = \{1..T\}$ be the index of item types, a packing *configuration* or *pattern*[1] is represented by vector $\mathbf{p}^h = (p^h_1, p^h_2, ...p^h_T) \in \mathbb{N}^T$, where h is unique pattern index. Gilmore and Gomory [1963] proposed a linear programming approach to determine the quantity of each pattern approximately, as defined by

[1]The term *pattern* is adopted in this thesis to highlight the reusability and knowledge-representation properties.

Eq.(2.8):

$$\min \mathbf{1} \cdot \mathbf{z}, s.t.\ \mathbf{Pz} \geq \mathbf{q}\ and\ \mathbf{z} \geq 0 \tag{2.8}$$

This linear programming formulation is known as *configuration LP*. The decision variable $\mathbf{z}$ signifies the quantity of patterns utilised in the final solution. Matrix $\mathbf{P}$ is an $H \times T$ matrix, where each column represents a valid pattern.

The optimal solution of the configuration LP, denoted as $\mathbf{z}^*$, provides a partial solution achieved by rounding $\lfloor \mathbf{z}^* \rfloor$. It is evident that at most T additional bins will be required to accommodate the fractional component. Nevertheless, despite being relaxed, the primary hurdle faced by configuration-LP-based algorithms remains the exponential count of feasible patterns. Karmarkar and Karp [1982] deliberated on the methodology of efficiently solving the configuration LP and employing rounding to produce an approximate solution through rounding and grouping. One significant outcome is that they achieved an absolute additive gap with the optimal solution. They assert the following statement for all $\alpha \in [0, 1]$:

Theorem 2.3 (Karmarkar and Karp [1982])**.** $R_{\mathcal{K}\&\mathcal{K}}(\alpha) = \mathcal{OPT}(L) + O(\log^2 \mathcal{OPT}(L))$

2.3 Online Bin Packing Problem

The online version of BPP requires making irreversible packing decisions upon item arrivals. This property prevents algorithm from accessing the full information of the item sequence, and solutions are constructed sequentially. Considering the formulation presented by Eq.(2.1)-(2.5), at step i, the items from $i + 1..N$ are unknown to the packing algorithm; in other words, the decision is made under incomplete information.

2.3.1 Metrics for Online Problem

The metrics discussed in Section 2.2 are also applied to analyse the performance of online algorithms. The metric *Competitive Ratio (CR)* [Tarjan, 1985] refers to the performance of some online algorithms compared to an offline optimal algorithm, where the entire item sequence is known in advance. The following gives the definition of Competitive Ratio:

Definition 2.9 (Competitive Ratio)**.**

$$CR_{\mathcal{A}} = \sup\left\{\frac{\mathcal{A}(L)}{\mathcal{OPT}(L)}\right\} \tag{2.9}$$

The Competitive Ratio is equivalent to the Asymptotic Performance Ratio when considering the asymptotic worst-case scenario [Coffman et al., 2013].

Researchers have also examined performance relative to the offline optimal solution on random inputs. There exists extensive discussion concerning the *regret* performance, defined as the additive difference between the online algorithm and the offline optimal solution.

Definition 2.10 (Regret of Online Algorithm)**.** $Re_{\mathcal{A}} = \mathcal{A}(L) - \mathcal{OPT}(L)$

Unfortunately, although both metrics assess algorithm performance, they are not equivalent measures, and no online algorithm can simultaneously optimise both metrics, as demonstrated by Andrew et al. [2013]. In comparison, CR represents a more conservative measure, while regret analysis focuses on average-case performance. Moreover, regret proves more suitable for experimental analysis when theoretical examination of CR proves intractable.

2.3.2 Online Algorithms with Zero Knowledge

There are many research works on online algorithms for BPP. Most classic algorithms do not assume any knowledge of the item sequence; these algorithms are referred to as "zero-knowledge" algorithms. The most well-known online algorithms are the Any-Fit family proposed by Johnson [1973], including Next Fit, Worst Fit, First Fit, and Best Fit discussed in the previous section. The Any-Fit property means not opening a new bin unless no existing bins can accommodate the current item. For First Fit and Best Fit, a stronger condition is satisfied: the current item will not be packed into the bin with the smallest content unless no other bin can accommodate the current item. These conditions ensure the opened bins maintain a high average utilisation level. Johnson [1974] proved that no algorithms satisfying the Any-Fit condition can break the 1.7 CR barrier. Lee and Lee [1985] proposed Harmonic family algorithms, achieving better CR through "reservation" techniques [Yao, 1980]. The bin interval is divided into sub-intervals according to a harmonic series. Each bin is assigned a specific interval, allowing only items whose size falls within that interval to be packed into the bin. The CR is proven to be:

Theorem 2.4 (Lee and Lee [1985])**.** $R^{\infty}_{\mathcal{HF}}(\alpha) \leq 1.691$

Although the reservation technique with a harmonic series has improved the lower bound, it may cause significant waste for large items with a size over $1/2B$. Several improved algorithms have been developed. Among them, Seiden [2002] achieved the best-known lower bound of APR at 1.592.

The above algorithms are designed based on the CR metric. In contrast, another group of researchers has discussed how the accumulated waste will grow under different distributions. [Courcoubetis and Weber, 1986] proposed three classes of distributions commonly adopted in waste regret discussions. Let the bin capacity be $B = 9$, and p_2, p_3 be the probabilities with sizes 2 and 3, respectively. Their

theorem is illustrated by following:

Theorem 2.5 (Courcoubetis and Weber [1986])**.** *Any discrete item-size distribution D falls in one of three categories based on the asymptotic growth rate of waste of the optimal offline algorithm $E[\mathtt{W}_D^{\mathcal{OPT}}(T)]$as a function of T.*

1. *Linear Waste (LW): $E[\mathtt{W}_D^{\mathcal{OPT}}(T)] = \Theta(T)$, e.g. $D = (p_2 = 0.8, p_3 = 0.2)$*
2. *Perfect Pack (PP): $E[\mathtt{W}_D^{\mathcal{OPT}}(T)] = \Theta(\sqrt{T})$, e.g. $D = (p_2 = 0.75, p_3 = 0.25)$*
3. *Bounded Waste (BW): $E[\mathtt{W}_D^{\mathcal{OPT}}(T)] = \Theta(1)$, e.g. $D = (p_2 = 0.5, p_3 = 0.5)$*

Csirik et al. [2006] developed the Sum-of-Square (SS) algorithm based on minimising expected regret. They introduced a simple heuristic that packs items to minimise the sum-of-squares potential of the packing configuration. The potential function is defined as $\sum_{l=1}^{B-1} \mathtt{N}p(l)^2$, where $\mathtt{N}p(l)$ denotes the number of bins with level l in the current packing solution. Their analysis demonstrated that under PP, the waste growth satisfies $E[\mathtt{W}^{SS}PP(T)] = O(\sqrt{T})$, while for BW this can be reduced to $E[\mathtt{W}^{SS}BW(T)] = O(\log T)$. However, in the LW case, the algorithm still exhibits linear waste growth: $E[\mathtt{W}_{LW}^{SS}(T)] = \Theta(T)$. More recently, Gupta and Radovanovic [2020] proposed an algorithm employing an online interior-point method, which achieves $E[\mathtt{W}^{\mathcal{G}\&\mathcal{R}}] = O(\sqrt{T})$ across all three distributions.

2.4 Online BPP with Half-Knowledge of Item Sequence

Online algorithms with zero-knowledge typically assume strict online characteristics and disregard any information about the item sequence. In contrast, many real-world scenarios neither adhere to such strict online conditions nor completely lack access to item sequence information. Previous analyses of offline and online

algorithms demonstrate that when complete item sequence information is available, algorithm performance typically improves. Consequently, researchers have developed algorithms that utilise partial information, which we collectively refer to as half-knowledge algorithms. These approaches address what we term the half-knowledge problem, which involves both relaxing online constraints and assuming some predictability of future item sequences.

2.4.1 Between Offline and Online

Problems exhibiting a degree of relaxation between online and offline characteristics are termed semi-online problems [Coffman et al., 2013]. These problems typically permit either constrained repacking of items or consideration of lookahead information about future items. A substantial body of literature explores various methods for relaxing traditional online constraints, enabling innovative approaches in scenarios where strict online conditions prove impractical or suboptimal.

Semi-online problems with repacking capabilities allow limited element repositioning. This concept was first examined by Galambos and Woeginger [1995] and Gambosi et al. [2006]. Ivkovic and Lloyd [1998] developed an advanced algorithm for fully dynamic bin packing using Harmonic-like classification, thoroughly analysing up to 30 potential packing patterns with their classifications. Their key innovation involved maintaining optimal combinations of large and small items through specific packing operations, achieving a competitive ratio of 4/3 that approaches the performance of offline algorithms like First-Fit Descending (FFD). More recently, Berndt et al. [2020] investigated reducing repacking operations by exploiting future information. They proved that for a competitive ratio of $1+\epsilon$, at most $O((1/\epsilon)^4 \log(1/\epsilon))$ repacks are needed, where ϵ denotes the migration factor. An alternative relaxation approach permits algorithms to examine upcoming items while packing current ones, or to temporarily hold items in a buffer before packing

decisions. The lookahead window or buffer size is typically bounded by parameter k to maintain the online nature of the problem. A basic strategy involves packing lookahead items using offline algorithms, as demonstrated by Grove [1995], who achieved a competitive ratio of 1.691.

Dunke and Nickel [2016] conducted comprehensive performance analysis for online problems with lookahead. For online Bin Packing Problem, they highlighted the efficacy and stability of short-range lookahead ($k \leq 15$). Their experiments revealed that exact re-optimisation provides only marginal advantages over rule-based lookahead algorithms.

2.4.2 Predcition-Based Algorithms

For the online BPP, a natural assumption is to predict the future incoming item sequence with a priori knowledge or online statistical learning. These approaches do not break the online decision constraint. However, these methods did not receive much attention from researchers until recently. Boyar et al. [2013] discussed an ideal model in which the online algorithm can access a tape of an offline oracle with unlimited computation power. They proved that with $\log n + o(\log n)$ bits of advice, the algorithm is able to achieve a CR of 1.5, while with linear advice, the CR can reach 9/8. Their analysis demonstrated the value of partially accessing the knowledge of the entire item sequence, thus somewhat relaxing the online constraint.

Angelopoulos et al. [2018] improved the competitive ratio to 1.47 with $O(1)$ advice for online algorithms. However, Antoniadis et al. [2020] pointed out the issue that if the advice includes errors, which are commonly observed in real-world problems, it can significantly decrease the Competitive Ratio. In some extreme cases, the Competitive Ratio can even reach 6 under fatal advice.

The works mentioned above highlight the importance of balancing the benefits and

risks of prediction. Building on this observation, Angelopoulos et al. [2023] studied online algorithms based on an offline solution of a partial item sequence given by First-Fit Descending rather than an offline oracle. This approach introduces the risk of giving erroneous advice. They also discussed a hybrid algorithm that balances the risk of error prediction by switching to zero-knowledge algorithms based on a step threshold. Lin et al. [2022] highlights the application of patterns with online learned item distributions. They designed packing patterns by analysis on the ratio of item size with bin capacity and used a Best-Fit Descending approach for planning the use of patterns. This work was later enhanced by the fuzzy bin selection method[Lin et al., 2024].

The distinction between prediction-based algorithms and semi-online algorithms with lookahead is not always clear-cut. A semi-online algorithm with lookahead can be viewed as a prediction-based algorithm that has perfect foresight of a short segment of the future item sequence. Generally, prediction-based algorithms are considered to have a longer prediction horizon, while semi-online algorithms with lookahead tend to be more short-sighted. Additionally, semi-online algorithms with lookahead do not account for prediction errors, whereas prediction-based algorithms often cannot avoid them.

2.5 Chapter Summary

This chapter provides a brief introduction to classical packing problems and their variants, including definitions of key elements such as bins and items, as well as classical formulations. It also discusses the distinctions between offline and online problems and their respective metrics. Based on accessibility to item sequence information, this chapter posits that classical offline problems possess complete information about the item sequence. In contrast, classical algorithms designed for online problems do not explicitly utilise item sequence information, categorising

them as zero-knowledge.

In particular, this chapter focuses on the intermediate area of half-knowledge problem settings and, based on an analysis of the existing literature, demonstrates that even the utilisation of partial information can often enhance the overall performance of algorithms.

Chapter 3

Literature Review

3.1 Methods of Solving COP

Traditionally, solutions for combinatorial optimisation problems were classified into four main categories: exact methods, approximation algorithms, heuristic algorithms, and metaheuristic algorithms. Exact methods, such as mixed integer programming and branch-and-bound techniques, could determine the theoretically optimal solution. However, they often incurred significant computational costs when addressing NP-hard challenges. Approximation algorithms provided guarantees about solution quality through theoretical validation, although they did not guarantee optimality. Heuristic algorithms swiftly identified feasible solutions but lacked theoretical underpinnings. Metaheuristic algorithms, including genetic algorithms and simulated annealing, offered broad search capabilities, yet solution quality varied depending on parameter adjustment. Each approach exhibited distinct strengths and weaknesses, necessitating a thorough evaluation of problem characteristics and solution requirements to determine the most suitable approach. In the realm of burgeoning machine learning advancements, particularly in deep learning methodologies, the concept of Learning to Optimise (L2O)

emerged [Bengio et al., 2021]. This innovative research domain is briefly outlined here.

3.1.1 Exact Methods

The exact algorithms for combinatorial optimisation problems were a series of algorithms aimed at finding the optimal solutions to these problems. The COP usually contained integer decision variables. Mathematically, problems with integer decisions involved were formalised as Mixed Integer Programming (MIP) and used solvers to solve them precisely. The MIP model was mainly solved with commercial solvers, which gave the best performance in terms of solution precision and time cost. In both academic research and industrial applications, CPLEX [Cplex, 2009] and Gurobi [Gurobi Optimization, LLC, 2024] were two commonly used commercial solvers that dominated the market share. Despite achieving excellent performance, these commercial MIP solvers were proprietary and closed-source software. Recently, several open-source MIP solvers also achieved reasonable performance for academic fields, including SCIP [Huangfu and Hall, 2018, Bolusani et al., 2024], HiGHS, and OR-Tools [Perron and Didier, 2024]. This thesis did not delve deep into the implementation details of integer solvers. However, this section provided an overview of the key algorithms of exact algorithms, which formed the backbone of these commercial or open-source MIP solvers.

The most commonly utilised exact algorithms included Branch-and-Bound (BnB) [Lawler and Wood, 1966], Branch-and-Price (BnP), and Branch-and-Cut (BnC). Branch-and-Bound was a systematic exploration algorithm that strategically reduced the search space by employing a divide-and-conquer approach and pruning methods to ultimately pinpoint the optimal solution. This technique utilised a tree search strategy to implicitly assess all potential solutions for a given problem, using pruning rules to discard areas of the search space that could not lead to an improved solution [Morrison et al., 2016]. Initially, the algorithm traversed the

entire search space, progressively narrowing down branches based on lower and upper bounds of likely solutions. By eliminating branches that were certain to produce suboptimal outcomes, Branch-and-Bound effectively reduced the search space, concentrating on more promising avenues. This recursive process persisted until the optimal solution was identified or all branches were exhausted. Branch-and-Bound was particularly effective for problems involving discrete variables and found common application in tasks such as integer programming and constraint satisfaction. The Branch-and-Bound algorithm revolved around three fundamental strategies [Morrison et al., 2016]: the search strategy, dictating the order in which subproblems within the tree were explored; the branching strategy, determining how the solution space was partitioned to generate new subproblems in the tree; and the pruning rules, which restricted the exploration of suboptimal regions within the tree. Literature such as Phan [2012], Vilà and Pereira [2014], Gendron et al. [2016] delved into the pruning approach incorporating estimated lower bounds. Recent success in machine learning also propelled research concerning the branching strategy [Alvarez et al., 2014, Scavuzzo et al., 2024]. Conversely, exploration strategies had not been the primary focus of recent research efforts.

Branch-and-Price and Branch-and-Cut were based on the idea of Dantzig-Wolfe Decomposition [Dantzig and Wolfe, 1960]. The Dantzig-Wolfe Decomposition was a mathematical programming technique that exploited the structure of subproblems to solve large-scale linear programmes. It decomposed the original problem into a restricted master problem and one or more subproblems. The master problem, an approximation of the original, iteratively expanded through column generation or row generation, improving the approximation. Subproblems supplied essential information, aiding the master problem's convergence to the original problem's optimal solution. This approach leveraged subproblems' tractability to efficiently solve complex integer linear problems.

The Branch-and-Price method was also recognised as column generation [Desrosiers

and Lübbecke, 2005]. The underlying concept was rooted in the observation that most columns were non-basic and that associated decision variables would be zero in the optimal solution. This allowed the exclusion of a significant portion of columns irrelevant to the problem's optimal solution. It solved a restricted master problem with only a subset of columns. A subproblem then identified columns that could improve the master problem's objective function. These columns were added iteratively. The advantage was that columns not generated by the subproblem were deemed non-improving, thus reducing the number of columns considered, enhancing efficiency. Branch-and-cut functioned as a complementary interpretation of Branch-and-Price, leveraging cutting planes [Gomory, 1958]. These cutting planes, constraints added to an integer programme, served to refine the feasible region without excluding any integer solutions. Initially, the search tree often commenced with an unconstrained or minimally constrained problem, gradually integrating constraints that enhanced the feasibility of the region. Hybrid algorithms that merged these strategies, such as Branch-and-Cut-and-Price, proved successful in addressing challenges like vehicle routing problems [Ceselli et al., 2021], bin packing problems [Wei et al., 2020], and cutting stock problems [da Silva and Schouery, 2023].

3.1.2 Approximation Algorithms

Approximation algorithms represented efficient computational tools that sought approximate solutions to hard optimisation problems, particularly those classified as NP-hard problems, while offering verifiable guarantees about the proximity of the obtained solution to the optimal one. These algorithms commonly emerged within the realm of theoretical computer science, stemming from the widely held conjecture that $P \neq NP$. This conjecture posited that a broad spectrum of optimisation challenges resisted exact resolution within polynomial time. The development and evaluation of approximation algorithms were closely tied to the

provision of mathematical proofs that confirmed the efficacy of the solutions returned, even under the most adverse circumstances. This distinctive characteristic distinguished them from heuristic approaches such as simulated annealing or genetic algorithms, which, although capable of yielding reasonably good solutions for specific inputs, lacked clear upfront guarantees about their potential for success or failure.

Chapter 2 examined several traditional approximation approaches for the bin packing problem, whether implemented in online or offline settings. A well-designed approximation algorithm could simultaneously achieve computational efficiency and guarantee performance bounds in worst-case scenarios. However, developing and rigorously analysing such algorithms required substantial effort, demanding considerable intellectual labour. Consequently, research efforts primarily focused on several fundamental combinatorial optimisation problems and their variants.

3.1.3 Heuristics

Heuristic algorithms provided feasible solutions for optimisation problems by generating candidate solutions and selecting the most promising one within reasonable computational constraints. While these solutions could not be guaranteed as optimal, they offered practical approaches when time efficiency was crucial. As discussed in Section 3.1.2, a fundamental distinction between heuristic and approximation algorithms was that the former did not require strict worst-case performance guarantees. In this sense, approximation algorithms could be considered a specialised subset of heuristic methods.

Among notable heuristic approaches, the A* algorithm [Norvig and Russell, 2021] became widely adopted for path-finding applications. This algorithm identified the shortest path between two points in a weighted graph, where edge weights were determined by a chosen heuristic function. The solution process aimed to minimise

the combined cost of path length and heuristic weight. The flexibility in heuristic function selection enabled the incorporation of domain-specific knowledge, such as obstacle avoidance, often yielding superior results to non-heuristic alternatives. Another significant example was the LKH algorithm [Helsgaun] for the Travelling Salesman Problem (TSP). Rather than using heuristic graph weights, LKH employed iterative improvement techniques. Starting with an initial solution, the algorithm applied edge-exchange operations to progressively refine the solution until reaching a satisfactory outcome. In this case, solution quality depended on carefully designed local search operators.

To reduce dependence on manually crafted heuristics, researchers investigated methods for automatic heuristic generation. This emerging field attracted growing interest from both the metaheuristics and artificial intelligence communities [Zhang et al., 2022, Romera-Paredes et al., 2024].

3.1.4 Metaheuristics

A metaheuristic was a sophisticated and problem-independent algorithmic framework that provided a set of guidelines or methodologies for developing heuristic optimisation algorithms [Kashyap and Mishra, 2022]. Metaheuristics were designed to address problems with extremely large solution spaces that were difficult to explore using conventional methods. As a result, metaheuristic research was closely linked to search methodologies. Unlike heuristics tailored to specific problems, metaheuristic frameworks could be applied across various problem classes, allowing their application to diverse industrial and scientific contexts. Metaheuristic algorithms represented a major research field, with numerous review papers focusing only on classification, such as those by Blum and Roli [2003], Raidl [2006], Glover and Sorensen [2015].

To summarise, metaheuristics could be categorised into local search-based meta-

heuristics, constructive metaheuristics, and population-based metaheuristics. Additionally, hyper-heuristics were also included. Unlike conventional metaheuristics that focused on the solution space, hyper-heuristics were search techniques employed for selecting, generating, and sequencing heuristics to tackle complex combinatorial optimisation problems.

Local Search

Local search, known as iterative improvement, sought effective solutions by iteratively adjusting a single solution, termed the current or incumbent solution. These adjustments, known as moves, were typically small to maintain proximity between neighbouring solutions based on a natural metric, hence the nomenclature of this metaheuristic class. The collection of solutions attainable by implementing a single move to a given solution was termed the solution's neighbourhood. Different move types could be established based on how the solution was represented, each defining a specific neighbourhood structure. During each iteration, the current solution was substituted with a solution from its neighbourhood. The approach employed to choose the new current solution was termed the move strategy or search strategy. A solution surpassing all others within its neighbourhood was identified as a local optimum, contrasting with a global optimum, which represented the best possible solution to the optimisation problem. Should the current solution reach a local optimum, a metaheuristic employed a strategy to surpass this point. This strategy distinguished a metaheuristic, often influencing its name. Local search metaheuristics mainly depended on iterative improvement to discover effective solutions.

A fundamental strategy of local search involved continually seeking a neighbour that outperformed the current solution, a method commonly known as hill climbing. Despite its simplicity, this approach was prone to being ensnared by local optima. Consequently, a significant portion of local search literature deliberated

on strategies to evade this local optima entrapment. One prominent method, simulated annealing (SA), was a probabilistic technique designed to approximate the global optimum of a given function. Drawing inspiration from metallurgical annealing processes, SA initiated at a high "temperature" and gradually progressed towards a superior solution space by occasionally accepting inferior solutions with diminishing probability, enabling it to escape local optima and approach the global optimum.

In contrast, Tabu search enhanced the search process by prohibiting the revisitation of certain solutions. By employing a tabu list in conjunction with local search procedures, Tabu search effectively navigated around local optima. This algorithm maintained a record of visited solutions in the tabu list, ensuring that neighbourhoods were only explored if they were not on the tabu list.

Variable Neighbourhood Search (VNS) centred on systematically altering neighbourhoods both during a descent phase to pinpoint a local optimum and in a perturbation phase to transition out of the corresponding valley. By incorporating diverse neighbourhood structures, VNS boosted the exploration diversity within the search process. These algorithms were commonly utilised in bin packing problems and their variations, with specific applications such as Tabu search for addressing 2-dimensional Bin Packing Problems (BPP), simulated annealing for 2-dimensional challenges focusing on packing rectangular items into circular containers, and variable neighbourhood search for BPP derivatives with intricate constraints.

These algorithms were actively applied for bin packing problems and their variants. Lodi et al. [1999] applied Tabu search for solving 2-dimensional BPP; Tole et al. [2023] applied simulated annealing for 2-dimensional problems, focusing on packing rectangular items into circular containers; Hemmelmayr et al. [2012], Meng et al. [2022] applied variable neighbourhood search for BPP derivatives with complex constraints including variable sized bins.

Constructive

Constructive metaheuristics, as the name implies, created solutions by assembling their fundamental elements instead of refining entire solutions. This method involved incrementally incorporating one element at a time into a partial solution. Constructive metaheuristics frequently stemmed from adaptations of greedy algorithms, where the most optimal element was added at each step. To enhance the quality of the ultimate solutions generated, the majority of constructive metaheuristics encompassed a local search phase following the construction phase.

Ant colony optimisation (ACO) [Mirjalili, 2019] was a metaheuristic inspired by ant foraging behaviour, which constructed solutions by simulating the pheromone trail mechanism. It used artificial ants that deposited pheromones on promising paths, guiding subsequent ants to optimal or near-optimal solutions. The algorithm iteratively updated pheromone levels, reinforcing good solutions while evaporating less effective ones. The ACO was commonly adopted in routing problems. Levine and Ducatelle attempted to apply ACO for bin packing problems. They proposed a hybrid algorithm that combined local search with ACO, showing excellent performance.

Large neighbourhood search (LNS) [Pisinger and Ropke, 2019], a variant of local search, could also be seen as the constructive counterpart of variable neighbourhood search. It worked by alternatingly destroying a solution and rebuilding it, usually using an extensive set of destroy and repair heuristics. Pisinger and Ropke [2007] proposed an improvement method by adaptively choosing destroy-repair heuristic pairs by their historical quality. This method was later known as adaptive large neighbourhood search (ALNS). Similar to VNS, LNS and ALNS were also widely adopted for the bin packing problem. Rodrigues et al. [2023] applied ALNS for generalised bin packing problems in last-mile delivery. Şafak and Erdoğan [2023] used LNS for container loading problems, an application of

BPP in logistics.

Population Based

Population-based metaheuristics achieved effective solutions through the iterative selection and combination of existing solutions within a designated set, commonly referred to as the population. Among these approaches, evolutionary algorithms (EAs) stood out as the most prominent as they emulated the fundamental principles of natural evolution. This dynamic field of study encompassed various techniques such as genetic algorithms (GAs) [Holland, 1992], genetic/evolutionary programming (GP/EP) [Koza, 1994], and evolution strategies (ES) [Hansen et al., 2015], all falling under the umbrella term of evolutionary algorithms.

Evolutionary algorithms functioned by operating on a designated set or population of solutions, utilising two key mechanisms to explore and identify optimal solutions: the selection of predominantly high-quality solutions from the population and the recombination of these solutions to generate new ones. This process involved specialised operators that melded the attributes of two or more solutions. Subsequent to recombination, these new solutions were reintroduced into the population, often subject to specific criteria like feasibility or minimal quality requirements, displacing other typically inferior solutions. The operators employed in evolutionary algorithms, namely selection, recombination, and reinsertion, heavily relied on randomness. Additionally, a mutation operator, which randomly introduced slight alterations to a solution post-recombination, was frequently incorporated.

In practice, most evolutionary algorithms repeated cycles of selection, recombination, mutation, and reinsertion multiple times, ultimately presenting the best solution within the population. To ensure the survival of superior solutions throughout the iterations while preserving population diversity, evolutionary algorithms typ-

ically necessitated some form of "population management." Particularly in the realm of COP, "pure" evolutionary algorithms were uncommon, often integrating enhancement operators, commonly in the guise of local search methodologies.

Evolutionary algorithms proved to be highly effective tools for optimising combinatorial problems, a fact underscored by their numerous successes [Stawowy, 2008, Kucukyilmaz and Kiziloz, 2018, Huang et al., 2022]. Notably, evolutionary algorithms demonstrated remarkable prowess in tackling multiobjective problems. In real-world scenarios, the need to address problems with multiple objectives—such as balancing risk and profit—became increasingly imperative. Liu et al. [2008] introduced evolutionary particle swarm optimisation for multiobjective bin packing problems. In logistics, the critical concern of achieving minimal packing while ensuring container stability inherently posed a multi-objective challenge. Fernández et al. [2013] explored the two-dimensional scenario with rotation, while Erbayrak et al. [2021] delved into the three-dimensional counterpart. Both studies leveraged multi-objective evolutionary algorithms to generate solutions.

While evolutionary algorithms dominated the landscape of high-quality research in population-based metaheuristics, alternative algorithms, often drawing inspiration from nature or metaphor, emerged. Noteworthy among these were successive algorithms like Ant Colony Optimisation (ACO) and Particle Swarm Optimisation (PSO) [Kennedy and Eberhart, 1995]. However, the proliferation of more recent nature- or metaphor-inspired metaheuristics began to face criticism within the research community for potentially masking a lack of originality behind intricate metaphors [Sörensen, 2015].

Hyper-Heuristics

Hyper-heuristics embodied search strategies crafted to choose, devise, and organise (meta)-heuristics aimed at tackling intricate optimisation challenges. The primary

goal was for hyper-heuristics to nurture the growth of more adaptable systems capable of handling a wide spectrum of problem domains, in contrast to prevailing meta-heuristic methodologies often customised for specific problems or limited categories of problems. The central focus of hyper-heuristics revolved around the astute selection of the most appropriate heuristic or algorithm for a given scenario. In a particular sense, a hyper-heuristic operated at a heightened conceptual level compared to the traditional application of meta-heuristics in optimisation problems. Essentially, a hyper-heuristic could be perceived as a (meta)-heuristic functioning on lower-level (meta-)heuristics. As a general heuristic search approach, hyper-heuristics typically strived to reduce the domain knowledge required for designing general heuristics. Due to their outstanding efficacy in multi-objective and machine learning-based optimisation settings, there was a burgeoning interest in the realm of hyper-heuristics.

Burke et al. [2006] employed genetic programming as hyper-heuristics, enabling the creation of an expression tree representing a heuristic for solving BPP, with low-level heuristics defined as mathematical operators and terminal variables. López-Camacho et al. [2014] introduced a comprehensive hyper-heuristics framework for offline 1- and 2-dimensional BPP, employing an evolutionary algorithm as a high-level metaheuristic to select the optimal heuristic from a pool of low-level heuristics. Asta et al. [2016] similarly utilised an evolutionary algorithm as a high-level strategy, aiming to develop a parameterised strategy matrix for heuristic selection, which guided the process of heuristic selection. Tu et al. [2023] applied reinforcement learning as a heuristic selector. These endeavours were categorised as selection hyper-heuristics according to the taxonomy outlined by Dokeroglu et al. [2024]. In contrast, generation hyper-heuristics sought methods to systematically generate a heuristic composed of low-level heuristics. Notably, a large language model was recently applied as a hyper-heuristic that evolved low-level heuristics [Romera-Paredes et al., 2024].

3.1.5 Learning to Optimise(L2O)

Machine learning is the research area that focuses on improving the performance of a system through experience or data [Mitchell, 1997]. It is a broad and vibrant research area closely linked to optimisation topics. Optimisation methods are utilised to help the learning system adjust model parameters based on data. This section briefly categorises the research field into supervised, unsupervised, and reinforcement learning.

Supervised Learning

In supervised learning, the system learned a mapping from input to output using input-output data [Russell et al., 2010]. The objective of the learning process was to minimise the gap between the model output and the example output label, known as the error. The measure that evaluated the gap between the model output and example output was the loss function, which varied according to the problem properties. Generalisation was one of the major concerns of supervised learning since researchers wanted the model to perform well on given examples and unknown future data.

Supervised learning was applied in several aspects of COP algorithms. One possible approach was to improve existing exact/approximate algorithms. Balteau-Lugojan et al. considered the exact solution of a non-convex quadratic programming problem. In this problem, supervised learning replaced the computationally expensive sub-matrix selection procedure. Several studies improved the performance of branch-and-bound methods to enhance the efficiency of branching [Alvarez et al., 2014, 2017, Khalil et al., 2016, Gasse et al., 2019].

Another approach was to learn a model to predict the solution, which could be viewed as an end-to-end approach. Vinyals et al. [2015] developed a model to

generate VRP solutions given a problem instance. Larsen et al. [2021] used a neural network to predict solutions for the stochastic loan programming model.

Finally, supervised learning was frequently used to help configure problems. Marković et al. [2005] built a stochastic customer demand model by training neural networks. Fuce and Wanghui [2010] learned to classify customer commands. Kruber et al. [2017], Bonami et al. [2018] applied learning to predict problem structures (whether to decompose or linearise) to improve solution speed. For real-world problems, Zhang et al. [2011] reviewed several data-driven traffic management systems where computer vision was applied to learn environments.

Unsupervised Learning

Unlike supervised learning, unsupervised learning did not have input-output pairs. Unsupervised learning aimed to capture the features of the examples. A common approach was to try to fit the joint distribution of given data. An example by Erdoğan and Miller-Hooks [2012] attempted to combine unsupervised learning to help solve the green-VRP problem. According to Bengio et al. [2021], there was very limited work that tried to combine unsupervised learning with COP.

Reinforcement Learning

Reinforcement learning was a framework that solved sequential decision problems [Sutton and Barto, 1998]. It trained an agent that could interact with a dynamic environment, obtaining the status of the environment (state), executing an action, and receiving feedback (reward) from the environment. The agent learned a policy to maximise the accumulated reward signal by trial-and-error search through interaction with the environment [Sutton and Barto, 1998, Nian et al., 2020]. In the operations research context, reinforcement learning was also referred to as approximate dynamic programming [Bertsekas, 2008] or neuro-dynamic program-

ming [Bertsimas et al., 2018].

Reinforcement learning (RL) is formalised as the optimal control problem of Markov decision processes (MDP) [Howard, 1960]. An MDP is defined by a 4-tuple (S, A, P_a, R_a) where S represents the state space, A represents the action space, $P_a(s, s') == \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the transition function between two states given an action and current state, $R_a(s, s')$ is a numerical reward signal for action a when state transfers from s to s'. The optimisation objective is to maximise the expectation of accumulated reward $E[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})]$. Under the Reinforcement learning literature, P_a and R_a are represented by the environment E. The agent is aiming to learn the policy $\pi : A \times S \leftarrow [0, 1], \pi(a, s) = \Pr(a_t = a | s_t = s)$ which could achieve the optimisation objective. For each state s_t, the agent picks up an action a_t for current state s_t, then the state of the environment will change to s_{t+1} accordingly, and the agent will receive a reward signal r_{t+1}.

On the other hand, deep learning (DL) was a wide set of machine learning methods based on artificial neural networks (ANN, or NN in short). The most significant difference between DL and other "shallow" machine learning methods was the application of deep and complex neural network structures, enabling the network to extract higher-dimensional features. In contrast, "shallow" machine learning methods usually used relatively simple models [Li, 2018]. Schmidhuber [2014] briefly described several combinations of RL and DL with relative papers. Deep learning could be used in modelling the environment, preprocessing RL input [Jodogne and Piater, 2007], modelling the value function of each state, or generating a policy [Sutton and Barto, 1998]. Important algorithms in DRL included deep Q networks [Mnih et al., 2016], policy gradient**-**based methods [Schulman et al., 2017], Actor-Critic [Mnih et al., 2016], and Monte Carlo Tree Search [Silver et al., 2016, 2017].

The research on DRL for BPP covered many RL approaches for 1D BPP and

could also be divided into two major mainstream approaches. One combined RL with heuristics/metaheuristics in a two-stage manner, and the other used RL only to model the policy of bin selection. Splitting the problem into two stages, that is, using a high-level hyper-heuristics model (where you might use the RL method) to pick a region or choose a heuristic, then generate a solution using low-level heuristics. This approach maintained the flexibility of heuristics while reducing the search space. Haffari and Shouraki applied a TD-λ based algorithm STAGE to learn the evaluation function to evaluate the quality of the initial state of local search and tested it on bin packing problems. Similarly, Silver et al. [2016] split bin packing problems into action regions and selected suitable heuristics by a hyper-heuristic framework. Recent research also showed that a single RL solution for BPP was possible with the development of RL. Cai et al. [2019] used reinforcement learning to generate an initial solution for further heuristics' optimisation. They also used the signal generated by heuristics to optimise the training of the RL model. Balaji et al. [2019] proposed using Proximal Policy Optimisation (PPO) to solve online stochastic optimisation problems, including BPP, VRP, and the news vendor problem.

The methods on higher dimension BPP were mixed according to what kind of problem the researchers were interested in. Hu et al. [2017] applied the policy-based network to produce an item packing sequence, given the original item sequence for 3D bin packing problems. The items were then packed using traditional heuristics. Wang and Nip [2017] used DRL to improve the resource assignment in Open-RAN networks, a practical problem which is highly related to BPP. Laterre et al. [2018] proposed a self-competitive mechanism to train an agent to beat the record generated by itself to solve 2D and 3D BPP. Kundu et al. [2019] applied Convolutional Neural Networks (CNN) [Zhao et al., 2024] to solve online 2D BPP. Verma et al. [2020] applied DQN to solve 3D BPP with only a single container. They defined movements for rectangle items and trained DQN to select movements. Zhao et al. [2021] used CNN to encode empty space for 3D BPP and then used constrained

programming to find the loading position. For multiple bins, they used the Monte Carlo permutation tree to adjust the sequence of item input and then placed items using the previously mentioned methods.

3.2 Patterns and Pattern Mining in COP

3.2.1 Forms of Pattern

Patterns represented powerful and highly effective problem-solving methodologies. Within the machine learning (ML) domain, the concept of pattern recognition was extensively employed to categorise entities based on their attributes. Specifically, within the ML community, pattern recognition predominantly centred on supervised or unsupervised learning [Jain et al., 2000]. These approaches typically entailed training adaptable classifiers with appropriately labelled data (supervised learning) or identifying significant patterns from unlabelled data (unsupervised learning).

The operations research (OR) community, dedicated to tackling combinatorial optimisation problems and adapting them to real-world scenarios, also maintained a longstanding interest in patterns and their recognition tasks [Nieddu and Patrizi, 2000]. Beyond the learning dimension, patterns were utilised to characterise prevalent structures found in high-quality solution frameworks or sets of rules that led to optimal solutions. Moreover, the OR community was closely intertwined with practical decision-making challenges, where problem setups might rely on mining historical data. This section offered a brief taxonomy of patterns in OR research literature and explored the shared conceptual ground of patterns between the two communities. The review of literature extended beyond the Bin Packing Problem (BPP) to provide a comprehensive overview of pattern and pattern-based research topics. It was essential to acknowledge the diverse interpretations of the

term "pattern" across various fields. For instance, within the ML community, studies might concentrate on classification, regression, and learning, topics closely linked to patterns and pattern recognition. In contrast, OR researchers might focus more specifically on the precise elements of solutions, such as bin configurations (refer to Section 2.2.3), columns, and routes. However, these specific elements fundamentally embodied patterns of solutions.

Component as Pattern

The most straightforward form of patterns was the components of a solution. This typically involved decomposing the solution of the problem into components and identifying components that frequently appeared in successful solutions. Essentially, patterns were utilised to construct solutions with improved objective values. In a more theoretical context, the processes of decomposition and reconstruction were described as set partitioning and set covering problems [Korte and Vygen, 2012]. The classical one-dimensional bin packing problem was one of the COP that constructed solutions based on patterns. In this problem, a method of packing items into a single bin was defined as a pattern. Some early work could be traced back to Gilmore and Gomory [1963]. Lin et al. [2022] constructed solutions for online BPP using analysis-based patterns. Marinelli et al. [2024] considered a combination of BPP and makespan minimisation problems. They adopted a similar pattern definition and jointly minimised bin usage and packing time.

Vehicle routing problems (VRP) were a widely studied COP family where pattern-based methods were employed. These types of problems involved planning a set of routes for a fleet of vehicles to visit various customers or locations, aiming to minimise the total distance travelled or the total cost of the routes. Rousseau et al. [2004] applied decomposition in VRP with time window constraints, where the pattern was defined as a set of customer nodes served by a single vehicle. They developed a column-generation-based approach that iteratively found route pat-

terns to enhance overall cost efficiency. Similarly, Stokkink et al. [2024] adopted a similar decomposition for the crowd-shipping problem, a more intricate VRP variant for last-mile delivery applications. Instead of considering the complete tour, they viewed the arcs of visited parcels as patterns. In addressing more complex real-world problems, such component patterns were utilised to provide groups of sub-decisions to circumvent low-level decision variables. An exemplary study was conducted by Cappanera and Scutellà [2015]. They grouped possible skilled operators' visiting timetables as patterns, aiming to assign patients based on the visiting pattern rather than determining individual patient-operator assignments. This definition of patterns reduced decision complexity and ensured solution quality through predefined visiting patterns.

Rule as Pattern

The second form involved patterns of rules. Instead of aiming for the optimal solution for a single problem instance with a fixed configuration, real-world applications typically necessitated systematic rules for solving a set of problem instances. Therefore, patterns of rules served as an inductive and efficient tool for summarising attributes and associated operations. In many instances, patterns of rules were problem-specific. Researchers aimed to identify effective rules or sets of rules to address a class of problems, often involving unknown problem instances. Interestingly, similar pattern mining topics were also explored in hyper-heuristics [Burke et al., 2013] and symbolic regression [Angelis et al., 2023].

Koonce and Tsai [2000] provided an example of how pattern-based methods could aid in deriving efficient heuristics by learning rule patterns from genetic algorithm solutions. They employed attribute-oriented induction to identify high-level patterns based on problem instance attributes and to develop algorithms using a set of rule patterns. Through pattern-based heuristics, they achieved superior results compared to the shortest process-time heuristics, thereby avoiding the need

to re-execute the genetic algorithm. Similarly, Barut et al. [2024] addressed the scheduling problem in a cloud computing scenario by mining patterns of rules. Their solution involved two stages: initially generating a pattern set of rules by mining historical solutions using K-means clustering, followed by constructing the solution by searching for combinations of rule patterns using particle swarm optimisation. The two-stage solution was also implemented in a semi-online manner by repeatedly adding the newly generated solution to the historical solution pool. By combining pattern-based methods and metaheuristic search, they significantly reduced the computational cost compared to directly searching the entire solution space.

In contrast, there were studies that mined patterns to assist general search algorithms. These patterns implicitly encoded problem attributes. From an analytical perspective, such patterns could enhance understanding of the problem and algorithms. Nezamoddini and Lam [2015] considered chromosome patterns in genetic algorithms to aid in network design. They proposed identifying reliable patterns throughout the evolution process and preserving well-behaved patterns. Experimental results showed that the pattern-guided genetic algorithm outperformed the pattern-free version. Tian et al. [2022] utilised patterns to enhance evolutionary algorithms, focusing on mining patterns for sparsely distributed Pareto optimal solutions in multi-objective optimisation tasks. The patterns were defined as masks of decision variables that made a significant contribution to achieving Pareto optimal solutions. Their results demonstrated that identifying sparse-variable patterns and applying rules to filter effective decision variables helped enhance solution quality.

Problem Configuration as Pattern

The third form of pattern involved patterns of problem configuration. With the increasing integration of big data in practical COP [Weinand et al., 2022], problem

parameters could be generated or estimated using a big data approach. In this context, pattern recognition was often viewed as a preliminary task in solving COP. Tong et al. [2023] addressed the taxi driver and customer matching problem using real-world application data, where greedy heuristics and traditional mathematical programming methods fell short. They employed reinforcement learning-based agents to assess the quality of optimisation results, with the optimal solution generated through an iterative optimise-evaluate process. Pourvaziri et al. [2024] combined machine learning for estimating waiting times and NSGA-II for planning electronic vehicle (EV) charging stations. The waiting time pattern was implicitly learned using neural networks. Their results demonstrated that hybrid methods could reduce waiting times by up to 61.5%. Additionally, Yang et al. [2024] explored the EV routing problem with a dynamic charging pricing strategy. They identified optimal charging pricing patterns using reinforcement learning agents. The pricing data was subsequently utilised by a genetic algorithm to make routing decisions.

3.2.2 Recognising Patterns in COP

The tools for pattern recognition tasks could be broadly categorised into three types: domain knowledge-based methods, where patterns were identified through analysis or expert-designed rules; data-driven methods, where the pattern mining process relied on appropriate data rather than explicit domain knowledge; and hybrid methods that combined domain knowledge and data-driven approaches.

Domain Knowledge-based Methods

Given the close relationship between COP research and operations research, it was crucial to develop patterns or pattern search rules using domain expert knowledge. One commonly used tool for incorporating expert knowledge was the fuzzy set

theory. The fuzzy set theory, introduced by Zadeh [1965], determined the degree of membership based on possibility values computed using membership functions. These membership functions could be tailored using domain knowledge [Fayyad et al., 1996]. For instance, Kagaya et al. [1994] utilised fuzzy set-based clustering techniques to identify trip patterns in demand-responsive transportation problems, which integrated vehicle routing and scheduling. Additionally, GRABOT and GENESTE [1994] proposed using fuzzy rules to consolidate dispatching strategies (such as shortest processing time or slack time) in job-shop scheduling problems. Recently, an illustrative example was provided by Lin et al. [2022], who devised patterns for the one-dimensional BPP by analysing the fraction of items relative to bin capacity.

For large-scale mixed linear programming problems, column generation, as introduced by Desrosiers and Lübbecke [2005], offered an approach to uncover vectors that formed linear combinations to address the original problem. These vectors could be conceptualised as patterns. Designing appropriate pricing problems or pattern generation problems and solving them typically necessitated expert knowledge. Zhu et al. [2012] developed a prototype column generation method for the multiple container loading problem. They relaxed constraints to generate patterns as prototypes, subsequently incorporating actual solutions into the pattern pool.

In a different domain, Bard and Purnomo [2005] explored methods for generating scheduling patterns for nurses based on various factors, including skills, costs, and preferences. They proposed a heuristic that maximised coverage during insufficient periods while considering preferences for scheduling pattern generation.

Addressing the flight-to-gate assignment problem in airport scheduling, Li et al. [2021] concentrated on solving patterns of flight-gate pairs. These patterns were generated using submodular approximation algorithms. While patterns could be generated using integer programming solvers, efficiency in handling large-scale instances remained challenging. Therefore, approximation and dynamic program-

ming heuristics were employed. For such problems, designing pattern generation (or pricing) algorithms typically required domain knowledge for the formulation and development of reasonable approximation heuristics.

Metaheuristics, known as higher-level procedures or heuristics designed to guide the search for solutions in optimisation problems with large solution spaces, aimed to find good solutions more efficiently than traditional methods. In COP research, these methods were widely applied in problems with complex constraints, rule mining, or multiple objectives that were hard to construct into expert solutions [Blum and Roli, 2003, Dhaenens and Jourdan, 2022]. Although designed to solve general problems, designing effective metaheuristics usually required expert knowledge due to the no-free-lunch theorem. The evolutionary algorithm was one of the most discussed metaheuristic algorithms that handled these challenges well. For example, cloud computing scheduling was a typical COP scenario where multiple resource constraints and multiple objectives were involved [Singh et al., 2022]. Gu et al. [2018] investigated feature selection in machine learning with metaheuristics, formalised as a COP. They applied the particle swarm optimisation (PSO) algorithm with threshold-based discretisation techniques to solve the problem. Sun et al. [2018] proposed a novel resource scheduling methodology using an improved Non-Dominated Sorting Genetic Algorithm II (NSGA-II). The performance metrics utilised were latency and stability. The authors enhanced the traditional NSGA-II by using a new formula for crowding distance. De Maio and Kimovski [2020] proposed a Pareto-based task offloading approach in fog for scheduling data-intensive scientific workflows by utilising a modified optimisation algorithm based on Non-Dominated Sorting Genetic Algorithm II (NSGA-II). An additional vector was used to uniquely mark dependencies between tasks and workflow levels, ensuring correct representation and evaluation of each individual. The proposed approach was evaluated using the Monte Carlo simulation framework. In other metaheuristic methods, Jacques et al. [2020] proposed using a dominance-based local search algorithm with a three-objective approach for medical data association

rule mining.

Data-Driven Methods

The data-driven methods for pattern recognition assumed that patterns could be revealed by mining high-quality data, thus relying less on domain experts. There were many successful applications of pattern recognition in computer vision [Astolfi et al., 2021], automatic driving [Qian et al., 2022], time-series analysis [Wen et al., 2023, Jin et al., 2024], etc. In the operations research field, the past decades also witnessed a rapid increase in interest in the discovery and analysis of business data. It was natural to adopt existing successful techniques, mostly machine learning, for recognising patterns in operations research and associated COPs. Bengio et al. [2021] classified how machine learning-based pattern mining techniques were applied to help solve COPs: end-to-end learning, learning to configure problems, and integrating with existing solution methods to assist low-level decision-making. The former two classes were major pure data-driven pattern mining approaches. End-to-end learning directly learned a strategy for solving COPs, where the solution strategy pattern was implicitly encoded in the learned models. Dai et al. [2018] proposed a general method to solve graph-based combinatorial optimisation problems based on graph embedding to encode a partial solution and reinforcement learning to learn a greedy policy. In contrast to the previous approaches, they used a value-based RL method, fitted Q-learning. Kool et al. [2019] proposed a transformer-based model to solve routing problems, such as TSP, CVRP, or split delivery VRP. Peng et al. [2020] generalised this approach to use a dynamic attention model so that state features could be updated during the construction of a solution. On the other hand, learning to configure problems attempted to use data-driven pattern mining techniques to predict the parameters. Section 3.2.1 lists several kinds of literature utilising machine learning as upstream parameter predictors. Xavier et al. [2020] provided an example of how

machine learning could help improve the problem configuration. They trained a K-nearest neighbour cluster to predict redundant constraints, good initial feasible solutions, and affine subspaces with a high probability of containing an optimal solution. These methods could significantly reduce the size of the problem. Bengio et al. [2021] extended the problem configuration to a broad sense, such that the initial states of combinatorial optimisation could also be included in problem configuration. Therefore, machine learning could be used to find initial solution patterns for downstream optimisation tasks.

Hybrid Methods

Many research works that combined expert knowledge and data-driven methods for solving COPs were proposed recently. The major motivation for such a combination was to utilise the benefits of both methods: the performance of expert knowledge was guaranteed, while data-driven methods excelled at identifying implicit patterns. One of the actively discussed topics was enhancing expert systems with data-driven methods. For example, Ismayilov and Topcuoglu [2020] integrated a neural network with a genetic algorithm to solve scheduling problems in cloud computing. The neural network was trained to predict the optimal solution in an unknown Pareto front. They applied the algorithm to Pegasus workflows and showed it to have advantages over other dynamic evolution algorithms. Huang et al. [2022] discussed how machine learning could help identify effective cuts for cutting plane methods, a powerful tool for solving mixed integer programming. They found that by injecting learning-based cutting pattern recognition, the cutting plane methods could achieve a 12.42% speedup without a loss of accuracy. Zhen et al. [2023] proposed a two-stage data-driven surrogate-assisted evolutionary algorithm for high-dimensional expensive problems (HDEP). They maintained a radial basis function to build a surrogate function of the global optimum. The surrogate was updated if a better fitness was achieved for each generation of evo-

lution.

In contrast, domain-knowledge-based methods could also enhance data-driven methods. Irwin-Harris et al. [2019] considered using an evolutionary algorithm to find good convolutional neural network (CNN) architectures. The architecture of the CNN was modelled as a directed acyclic graph, allowing nonlinear network architectures to be investigated. Zhang et al. [2021] systematically analysed the sparsity of the feature matrix for the feature selection task in machine learning. They built an integer programming model and optimised it using the alternating direction method of multipliers (ADMM). Liu et al. [2023] provided a comprehensive review. In general, these hybrid research works combined domain knowledge-based and data-driven methods in an upstream-downstream pattern.

3.3 Variances on Bin Packing Problems

3.3.1 Variances on Bins

The classical configuration of BPP defined the bin as a trivial container that could pack any kind of item. When considering scenarios more related to the real world, the configuration of bins could vary according to specific requirements. This section provides a brief introduction to problems with different bin configurations and the related algorithms.

In some scheduling scenarios, the maximum number of open bins was constrained. Such a problem was termed *k-bounded* BPP, where k represents the maximum number of open bins. Algorithms that adhered to the bin number constraint are referred to as bounded-space algorithms [Coffman et al., 2013]. To prevent the violation of k, a closing rule was necessary to determine which bin to close. Early literature discussed two simple closing rules: First Close, where the bin with

the smallest index was closed, and Best Close, where bins with the highest level were closed. The discussion involved the interplay of packing rules and closing rules. The abbreviation AXY_k was used to denote this combination, where X represented the packing rule (F for FF and B for BF), and Y represents the closing rule (F for First Close and B for Best Close). Mao [1993] and Zhang and Yue [1997] discovered that $R^{\infty}_{\mathcal{ABF}_{\|}} = 1.7 + 3/10k \geq R^{\infty}_{\mathcal{AFB}_{\|}} = R^{\infty}_{\mathcal{NF}}$ when $k \geq 2$, but the competitive ratio had a remainder term linked to the bound k. Csirik and Johnson [2001] demonstrated that the asymptotic behaviour of ABB_k was exactly the same as FF and BF, without the k-related remainder term. Woeginger [1993] achieved a competitive ratio of 1.694 with the Simplified Harmonic algorithm for $k \geq 6$, but no better results were reported for $k < 6$. A common trend was that increasing k allows for the development of algorithms with better performance, but designing the algorithm with the minimum number of opened bins remains an open research question.

Another actively discussed variant was Variable Sized Bin Packing Problem (VSBPP), where the capacity of bins could vary. For each capacity type, an infinite number of bins was available. The objective might also involve minimising the number of bins. Essentially, the objective was to minimise the total bin cost. Csirik [1989] found an interesting result that with two bins of different sizes, it was possible to achieve better worst-case performance than the classic BPP configuration with the Harmonic strategy. For multiple bin sizes, Kang and Park [2003] discussed an algorithm that iteratively packed items with FFD or BFD and updated the packing by swapping the bin with the lowest level with a low-capacity one. The optimal competitive ratio of 11/9 was attainable when item sizes and bin capacities were divisible; otherwise, they achieved 3/2. A more general configuration involved assigning different costs to bins. Crainic et al. [2011] presented heuristics also based on BFD, where bins were arranged by the ratio of cost to bin capacity. In summary, most researchers extended existing methods from classical BPP to VSBPP. However, it was also noted that VSBPP or its

variant might arise during the online execution of classic BPP, where bins might be partially filled. Exploring whether research outcomes in VSBPP could enhance the classical problem presented an intriguing problem.

3.3.2 Variance on Packing

The classic bin packing problem involved allocating items into bins in a straightforward manner. Driven by real-world applications, predominantly in management issues within operating systems and cloud computing, researchers explored various scenarios where the packing rule might vary according to practical considerations.

When the deletion of items was permitted at each step, the problem evolved into the Dynamic Bin Packing Problem. The objective shifted to minimising the upper bound of bin usage throughout the execution horizon, closely linked to dynamic resource allocation. Coffman et al. [1983] was the first to study this problem using two different variants of FF, demonstrating a competitive ratio between 2.25 and 2.89 with a maximum item size ranging from $B/2$ to B. In the context of cloud computing, the significance of the last time resources (bins) were occupied was emphasised. Li et al. [2016] and Ren et al. [2017] explored the traditional Any-Fit family in dynamic BPP for on-demand cloud resource allocation, where each bin incurred a continuous cost unless it was empty. De Cauwer et al. [2016] investigated a similar configuration by treating the problem as offline and employing MIP for its solution.

Another variation studied by Dósa and He [2006] was based on remote file system management. In this scenario, not all items were packed; those that remained unpacked incurred "rejection" costs. The objective was to minimise the sum of the number of bins and rejection costs. They examined both offline and online versions using modified Any-Fit algorithms, achieving an absolute worst-case ratio of 2 for the offline problem and a competitive ratio of 1.75. A subsequent work by

Epstein and Levin [2010] aimed to enhance the time complexity of execution by grouping small items. This was later enhanced by Epstein [2010] with Harmonic-Fit, achieving a competitive ratio of 538/333.

The study on cardinality constraints was also motivated by discrete application problems. In the multiprocessor scheduling context, where the maximum number of tasks that could be executed was restricted, modelling with BPP involved limiting the maximum quantity of items in a bin, known as a cardinality constraint [Krause et al., 1975]. Kellerer and Pferschy [1999] demonstrated a lower bound of 3/2 and an algorithm with a running time of $O(n \log^2 n)$. Balogh et al. [2020] investigated a series of tighter lower bounds for different item number limits.

Lastly, the selfish bin packing problem considered packing carried out by self-interested decentralised agents. In this setup, each item sought placement with the lowest cost $s/\sum_i s_i x_{ij}$, aiming to find a bin that was as full as possible. By reaching a Nash Equilibrium, where no item could improve its placement, the global objective of minimising bin usage could be achieved. Performance in selfish bin packing was evaluated using the Price of Anarchy (PoA), which denoted the upper bound of the ratio between the selfish and optimal solutions. Yu and Zhang [2008] proposed an algorithm with a time complexity of $O(n^4)$ that attained the Nash Equilibrium with a PoA between 1.64 and 1.66. Zhang and Zhang [2022] explored a discount-sharing mechanism where larger items bore higher costs than smaller items, resulting in a PoA of 22/15 or less. Ye et al. [2022] analysed the performance of trustworthy mechanisms for selfish agents in a cloud computing context. Their algorithm, tested using Amazon and Google Cloud data, achieved a PoA ranging from 1.17 to 1.38 for offline 1D problems.

3.3.3 Multi-Dimensional BPP

Consider a common scenario in logistics: commodities needed to be transported using standard containers in modern logistics. To eliminate transportation costs, the space of containers had to be used effectively. Such a typical scenario also appeared in the manufacturing industry, cloud computing, and scheduling [Lodi et al., 2002, Feng et al., 2019, Cid-Garcia and Rios-Solis, 2020]. This problem was named the multi-dimensional bin packing problem, a natural extension of the classic BPP. The volumes of items and bins were represented by vectors with at least two dimensions. The multi-dimensional BPP was also NP-hard: when the items only differed in one component, the multi-dimensional version degenerated to the classic problem.

The problem configuration of multi-dimensional BPP could be classified according to different aspects. Firstly, the dimension difference could result in different solution schemes, analysis techniques, and application scenarios. The 2-dimensional and 3-dimensional BPP received the most attention in the multi-dimensional bin packing problem, while more than three dimensions were less discussed. Secondly, the multi-dimensional BPP could be divided into geometry bin packing and vector bin packing [Christensen et al., 2017] by type of constraint. The geometry bin packing problem treated the items and bins as geometric objects, such as rectangles. An item had to be packed into a bin satisfying the geometric non-overlap and containment constraints. In contrast, in the vector bin packing problem, the items packed into a bin followed the vector addition rule and were subject to component-independent capacity constraints. Additionally, when some of the dimensions did not have volume constraints but the geometric constraints were still required, the problem was described as strip packing. 2-dimensional strip packing was widely discussed in the context of planning and scheduling. Finally, the orthogonal configuration of the multi-dimension bin packing problem treated the bins and items as rectangular. Driven by the practice in logistics and manufacturing, researchers

proposed irregular packing problems where the items had non-rectangular shapes. This configuration further increased the hardness of geometric constraints.

The most well-studied orthogonal configurations of multi-dimensional BPP were 2-dimensional and 3-dimensional geometric bin packing with rectangular items and bins. They performed differently compared with the classic 1-dimension BPP. Determining whether a set of items could be packed into one bin was NP-hard, even if the item was square [Leung et al., 1990]. This was the most critical issue of geometric bin packing: identifying a valid bin for the given item was hard. Baker et al. [1980] proposed the Bottom-Left algorithm by letting the item be sorted by width, starting packing at the lowest possible position and left justified. Chung et al. [1982] introduced a 2-phase algorithm that obtained a strip packing by Hybrid First-Fit and then considered the strip as an item to solve 1D BPP. 2-phase heuristics were further enriched and discussed by Frenk and Galambos [1987] and Berkey and Wang [1987]. Coffman et al. [2004] extended 1D heuristics First-Fit Decreasing and Next-Fit Decreasing to offline 2D problems as FFD-Height and NFD-Height and analysed the worst case, later Best-Fit Decreasing was also analysed. These methods considered the height dimension of the 2D item and justified the item on the left in the strip packing problem. The most advanced offline approximation algorithm for 2-dimensional BPP was given by Bansal and Khan [2013] with APR of 1.405. For online 2-dimensional problems, Epstein and van Stee [2006] achieved an improved CR of 2.25, whereas the offline best APR was 1.405 by Bansal and Khan [2013] when rotating an item was allowed. Han et al. [2011] proved the CR upper bound of online 2-dimentional problem was 2.554. For 3-dimensional problems, Caprara [2008] achieved the best known offline APR of 2.86, while Han et al. [2011] claimed their algorithm achieved APR of 4.31 in online 3-dimensional problems.

Similar to the 1-dimensional BPP, most studies on variations of the multi-dimensional BPP were motivated by practical applications. For instance, in cloud computing

and scheduling, where explicit geometric constraints were absent, a cloud service provider had to ensure that resources allocated to virtual machines, such as memory, storage, and CPU, did not surpass predefined limits. Properties like memory, storage, and CPU for a particular virtual machine were interdependent concerning resource allocation. Such scenarios were often modelled as vector bin packing, where each component of an item vector represented resource demands [Sheng et al., 2022]. The general vector bin packing problem was NP-hard [Woeginger, 1997]. The best-known competitive ratio for arbitrary dimensions was $d + 0.7$, as demonstrated by Garey et al. [1976] by employing the FF algorithm in vector bin packing. However, a special case was highlighted by Christensen et al. [2017]: defining a partial order of vectors enabled APTAS for the problem. For arbitrary dimensions, Bansal et al. [2009] achieved an APR of $1 + \log d$ using a round and approximate framework, albeit with exponential complexity relative to the number of dimensions. Studies on online vector packing were diverse, covering various configurations such as 0-1 vectors [Azar et al., 2013] and small vectors [Azar et al., 2015].

Strip packing could be seen as a fusion of bin packing problems and makespan minimisation problems. Many planning and scheduling issues could be framed as strip packing dilemmas, such as representing Gantt charts as 2-dimensional strip packing challenges. Several studies concentrated on developing absolute worst-case approximation schemes concerning practical planning problems. Gálvez et al. [2018] devised an absolute approximation of $4/3 + \epsilon$ within pseudo-polynomial time constraints. For asymptotic analysis, Coffman et al. [1980] extended the FF algorithm with decreasing height for 2-dimensional strip packing, achieving an APR of 1.7, a result later refined by Golan [1981] and Baker et al. [1981]. Han et al. [2016] drew an intriguing conclusion regarding online asymptotic analysis for 2-dimensional strip packing, indicating similarities with the classic 1-dimensional BPP in terms of competitive ratio.

The irregular bin packing problem involved handling non-rectangular cases, which posed challenges for approximation analysis. Addressing this issue often required employing metaheuristics and (mixed) integer programming models. Geometric analysis tools like polygons [Burke et al., 2007] and trigonometry were commonly utilised to tackle irregular packing problems. Due to the complexity of analysis, existing studies typically provided only weak lower bounds [Leao et al., 2020]. However, the lack of theoretical assurances did not hinder its application in a broad spectrum of industrial challenges [M'Hallah and Bouziri, 2016, Qin et al., 2018, Griffiths et al., 2019]. The ongoing research trend in irregular packing emphasised both theoretical analysis and the attainment of effective solutions within constrained computational resources for practical datasets [Guo et al., 2022].

Given the primary focus of this thesis on the 1-dimensional case, the discussion on multi-dimensional problems is concise, providing a broad overview of related areas. For interested readers, several surveys are recommended, such as those by Wäscher et al. [2007] offering a systematic typology of the bin packing problem family. Works by Christensen et al. [2017] delved into geometric bin packing problem approximations. Christensen et al. [2017] also extended the discourse to the vector bin packing problem. For irregular geometric problems, studies by Leao et al. [2020] and Guo et al. [2022] focused on the 2D version. Additionally, Ali et al. [2022] concentrated on online 3D geometric problems, exploring a variety of approximation algorithms, metaheuristics, and related techniques. Notably, Wu et al. [2023] reviewed works reflecting the mounting interest from the machine learning community.

3.4 Chapter Summary

This chapter provided a broad review, analysis, and discussion of pattern-based approaches for solving COP, as well as various variants of BPP.

In terms of solution methodologies, this chapter reviewed classical approximation algorithms and heuristics. The key distinction between the two was that the former often came with rigorous worst-case performance guarantees, whereas the latter were typically designed based on empirical insights. For large-scale problems with more complex constraints, the metaheuristic family, as a representative class of problem-free, search-based methods, was employed to obtain high-quality approximate solutions within limited computational resources. Based on population classification, single-population-based search methods were primarily divided into local search and constructive approaches, while multiple-population methods were largely composed of evolutionary algorithms and their variants. When the search space was extended to the heuristic level, hyper-heuristics were introduced to generate rules and algorithms for solving COP rather than merely obtaining solutions to specific problem instances. Finally, with the rapid advancements in machine learning in recent years, the learning-to-optimisation paradigm was proposed and applied to solving COP. This chapter also reviewed relevant developments in this area.

Next, this chapter examined patterns and pattern mining methods in COP. Due to the structural diversity of COP problems and the variety of challenges they addressed, different types of patterns used in COP solving were reviewed. These primarily included problem components as patterns, which formed part of the solution; heuristics or rules as patterns, which operated at a higher level; and problem configurations as patterns, which defined overarching structural properties. Regarding pattern mining methods, COP problems often incorporated rich domain knowledge, leading to significant diversity in mining techniques. Three major categories of approaches were covered: domain-knowledge-based methods, data-driven methods, and hybrid methods that integrated expert knowledge with data-driven techniques.

Finally, this chapter delved into several variations of bin-packing problems and

their theoretical approximation analyses. These variations encompassed various bin and item arrangements. The study of multidimensional BPP was also carried out, highlighting its critical distinctions from the traditional one-dimensional configuration.

For interested readers, several bin-packing problems surveys were recommended for further research. For online problems, Galambos and Woeginger [1995] reviewed several heuristic-based approximation methods. For general topics of Bin Packing Problem, Delorme et al. [2016] presented the definition, approximation and exact methods of 1D BPP. Lodi et al. [2002] focused on 2D problems and Martello et al. [2000] reviewed 3D problems. Christensen et al. [2017] focused on approximation methods including heuristics and metaheuristics algorithms.

Chapter 4

A Plan-and-Pack Framework

4.1 Introduction

When people plan to travel, they often need to make advance arrangements such as booking hotels, planning travel routes and transportation, and preparing luggage. These plans are typically made based on prior knowledge of the travel destination and relevant information. However, during the actual journey, there are often factors that can affect the execution of these plans. These factors may be negative, such as delays in booked transportation or closures of attractions, or they may be positive, such as discovering more interesting sights along the way or being able to alter tickets with an earlier departure time. Therefore, when faced with these factors, travel plans are often adjusted during the journey. In general, one tends to maintain the overall travel plan while making minor adjustments to adapt to new circumstances. This example illustrates a simple but effective decision-making principle: when dealing with noisy information brought by external randomness, plans can still be made based on this information. Adjustments can then be made during the execution of the plan based on newly observed information. Inspired by the above idea, this chapter attempts to introduce a similar decision-making

approach for the OBPP, which also involves external stochasticity.

For the OBPP, the major difference between prediction-based methods and zero-knowledge methods is that the prediction-based approach enables proactive planning. Through prediction, it becomes possible to solve an offline problem that closely approximates the real online problem. The analysis in Chapter 2 has already demonstrated that solving offline problems for the same issue often results in higher-quality solutions. Efficient patterns can be discovered from the solutions of these offline problems and reused, thereby providing a template for online solving and guiding online packing to find better solutions. Specifically, there is a two-stage framework which consists of two parts: firstly, planning how to pack items for a finite number of future steps, known as the planning stage; secondly, actually executing the packing process under the guidance of patterns, which we refer to as pattern-based packing. The entire framework is named Plan-and-Pack. Here, patterns and plans based on patterns serve as a bridge for extracting implicit knowledge and connecting knowledge to decision-making. We will present the entire Plan-and-Pack framework in Section 4.2, with this chapter focusing on discussions at the framework level. Detailed implementation of the plan-and-pack framework will be introduced subsequently.

4.2 Plan-and-Pack Framework Overview

4.2.1 Framework overview

The proposed CGPP framework is shown in Figure 4.1. The algorithm comprises three main stages: distribution estimation, plan generation, and packing. The distribution estimation module uses a short-term sequence memory to estimate the real-time distribution of random variables if the information is not known. The blue rectangle represents the plan generation procedure, which applies the

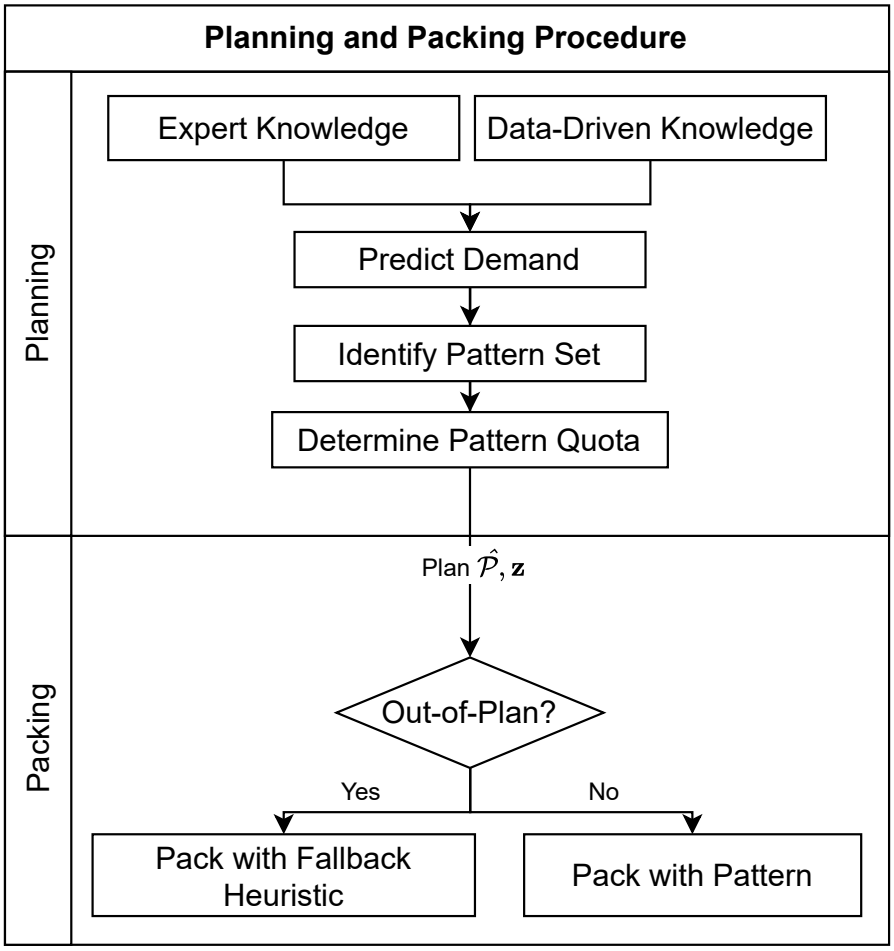


Figure 4.1: The plan-and-pack framework.

dualism pricing method to identify good patterns and generate a plan to guide future packing. The yellow rectangle describes the adaptive packing process in CGPP under the guidance of the packing plan and includes fallback strategies in the event of poor estimation errors. Details of this process are described in the following few subsections.

4.2.2 Generate Patterns with Prediction

Formally, in the online BPP, it is assumed that a problem instance is a finite sequence of items of length N, with index $i = 1, 2, ..., N$. Each item belongs to a finite type $t = 1, 2, ..., T$, which is associated with a size s_t. The quantity of item type t in the sequence is defined as q_t, and its value is determined by sampling from

a given distribution D. In practice, the stochastic process of items could be more complex in the sense that the distribution could change over time. In this case, it becomes a non-stationary distribution problem that is more difficult to solve. Both stationary and non-stationary distributions of problems are investigated in this study.

Let the pattern be represented by a vector of the quantity of all item types that can be packed into a bin, i.e. $\mathbf{p}^h = (p_1^h, p_2^h, ..., p_t^h, ..., p_T^h)$. Denote $\mathcal{P}$ to be the set of all feasible patterns, as Eq.(4.1):

$$\mathcal{P} = \mathbf{p}^h | \sum_{t=1}^{T} p_t^h s_t \leq B, p_t^h \in \mathbb{N}, h = 1, 2, ... \tag{4.1}$$

where B is the capacity of bins, s_t is the size of item type t.

A pattern generation integer programming problem is defined by reformulating the original integer programming formulation of BPP by Eq.(2.1)-(2.5):

$$\min \sum_{\mathbf{p}^h \in \mathcal{P}} z^h \tag{4.2}$$

$$s.t.$$

$$\sum_{\mathbf{p} \in \mathcal{P}} p_t^h z^h \geq q_t \qquad \forall t \in \{1, .., T\} \tag{4.3}$$

$$z^h \in \mathbb{N} \qquad \forall h \in \{1, 2, ..\} \tag{4.4}$$

The decision variable z^h signifies the quantity of pattern p_h utilised in the final solution. Since searching all feasible patterns is not acceptable in most cases, a subset $\hat{\mathcal{P}} \in \mathcal{P}$ that is able to satisfy all demand q_t is always chosen. By solving the newly defined problem of Eq.(4.2)-(4.4), one can get the packing plan. The packing plan involves the subset $\hat{\mathcal{P}}$ and its associated quota $\mathbf{z}$.

It is obvious that determining the subset $\hat{\mathcal{P}}$ can significantly affect the quality of planning. For example, a trivial pattern set where each pattern involves only one

type of item will result in a huge waste of space. Therefore, determining a subset with efficient patterns is critical to the planning quality.

Assuming predictions can achieve perfect accuracy, i.e. the exact quantities of items to be packed are known, the online bin packing problem degenerates into an offline problem. In general scenarios, predictions are based on expert knowledge or obtained through data-driven approaches like online learning. Such methods will likely have prediction errors. These errors often lead to the imperfect execution of the generated packing plans. Methods for generating patterns and associated packing plans typically involve solving offline bin packing problems, either exactly or approximately.

4.2.3 Pattern-Guided Packing

To solve the problem defined in Section 4.2.2, one can obtain a pattern set $\hat{\mathcal{P}}$ along with its corresponding usage quantities, which will be used to guide online packing. Specifically, when opening a bin, a pattern will be assigned to that bin. This pattern will describe the items that might be packed into this bin in the near future. When an item arrives, the system will always first look for an open bin that matches the pattern for packing that specific item. If a suitable bin with a matching pattern is found, the item will be packed into that bin. Otherwise, a new bin will be opened and assigned a new pattern. Within the planning scope, the usage quantity of each pattern with index h will not exceed its planned quota z^h.

Considering the presence of prediction errors, the actual quantity of items may be more or less than predicted, which can affect the execution of the packing plan. Specifically, the plan may not be able to accommodate the excess items when there are more items than predicted, meaning that packing those items may not benefit from the predefined plan. In contrast, some patterns in the plan may only

be partially packed when there are fewer items than predicted. Both scenarios can lead to a decrease in performance. Therefore, during the packing process, it is necessary to apply a dynamic packing strategy rather than strictly adhering to the existing plan.

4.3 Feasibility Analysis of Plan-and-Pack Framework

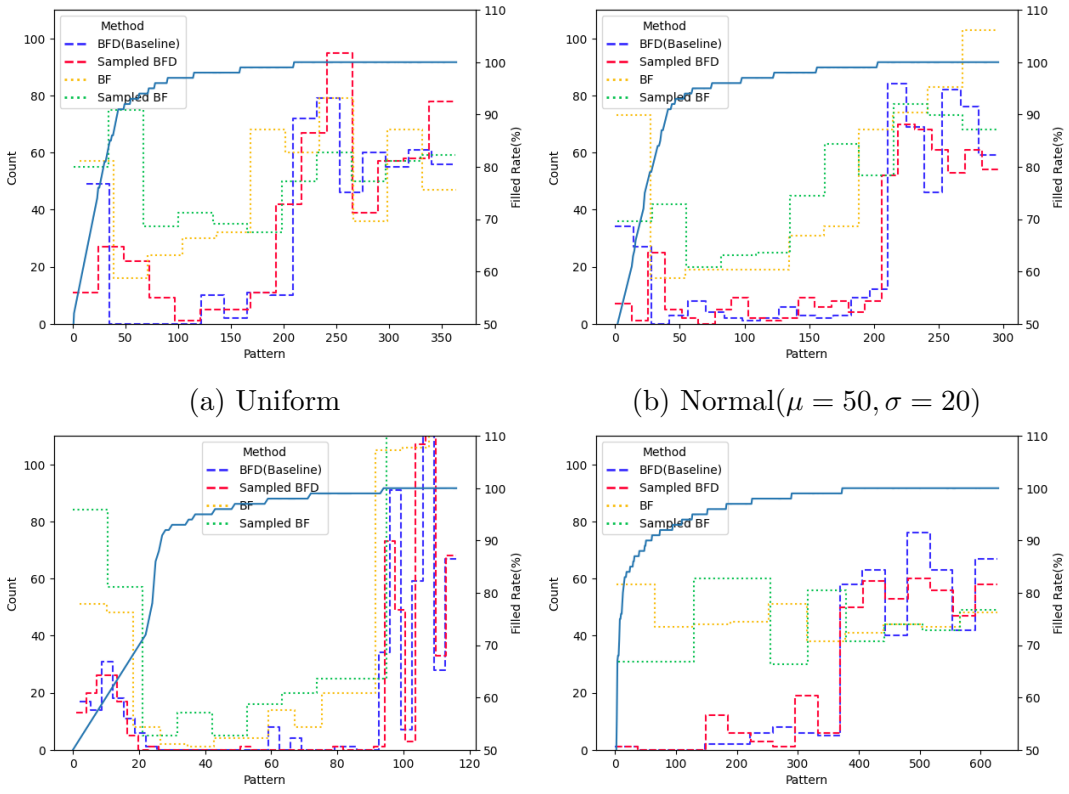
4.3.1 Mining Patterns under Stochastic Environment

In practical scenarios of the OBPP, where actual item demand remains unknown, pattern mining must operate on demand estimates. While this inevitably introduces prediction errors, our methodology nevertheless prioritises the extraction of robust packing patterns. To validate this approach, two controlled experiments with constrained parameters are presented: bin capacity $B = 100$, item types $1, 2, ..., 100$ and item sequence with length 1000. The packing patterns are obtained by solving the bin packing problem instantiated by the generated item sequence. Crucially, each unique bin configuration in the resulting solution is treated as a distinct pattern, with its recurrence frequency directly determining the associated quota value $\mathbf{z}$.

The first experiment considers an independent and identically distributed (i.i.d.) scenario, incorporating four common distributions: Uniform, Normal, Poisson, and Weibull. For each distribution, two item sequences are generated: a target sequence (representing the actual operational demand) and a sampled sequence (serving as a distributionally equivalent approximation). Both sequences are solved using the Best Fit and BFD algorithms, with the BFD solution for the target sequence acting as the baseline. Figure 4.2 displays the distribution plots

for all four cases, where solid blue lines denote the pattern filled rate (total item volume per pattern as a percentage of bin capacity). Two kinds of dashed lines differentiate the pattern quantities obtained via BFD and BF.

The results demonstrate a consistent divergence between BF-derived patterns and the baseline across all distributions. In contrast, BFD applied to the sampled sequence yields pattern distributions closely aligned with the baseline. Notably, BFD consistently achieves higher filled rates than BF. This experiment highlights two critical observations: Firstly, high-quality solutions for approximately sampled sequences can approximate the quality of offline solutions for target sequences. Secondly, basic online algorithms like BF fail to match this performance even when applied to identical sequences. These findings underscore BFD's robustness in mitigating prediction errors inherent to demand estimation. The second



(a) Uniform

(b) Normal($\mu = 50, \sigma = 20$)

(c) Poisson

(d) Weibull

Figure 4.2: Pattern Distribution with Different Methods

experiment employs an error-injected uniform distribution to simulate imperfect

distribution forecasting. Specifically, each item type's occurrence probability is defined as $p(t) = 1/T + \epsilon$, where T denotes the total item types and $\epsilon \sim \mathcal{N}(0, \sigma)$. The standard uniform distribution corresponds to $\sigma = 0$, with increasing σ values inducing progressive deviation from uniform distribution. Uniform distribution sampling is maintained for the target sequence, deriving optimal offline solutions via BFD execution on this sequence.

The perturbed distributions are generated across $\sigma \in [0, 0.02]$ with 0.005 increments, independently sampling five instances per σ for pattern mining. A method's efficacy is quantified by its ability to approximate the ideal solution's distribution. Consequently, performance is measured through the quota of shared distribution in the target sequence's BFD solution. Four approximation algorithms are evaluated: Best-Fit Descending, First-Fit Descending, Best Fit, and First Fit, with results compared against a baseline established by executing Best Fit directly on the target instance.

Figure 4.3 presents the experimental results. Overall, offline algorithms (BFD, FFD) applied to error-perturbed samples yield pattern distributions closer to the ideal than online algorithms (BF, FF). Among offline methods, BFD and FFD exhibit comparable performance. For online approaches, BF generates patterns more akin to offline solutions than FF does. Notably, at lower error levels, BFD and FFD produce higher-quality patterns than those obtained by executing BF directly on the target sequence. Even at substantial error magnitudes ($\sigma > 0.01$), both offline algorithms retain approximately 30% common patterns with the baseline. This demonstrates that mining patterns with high-quality solvers can achieve superior performance under moderate prediction errors. It is also worth noting that each item type has a probability of $p(t) = 0.01 + \epsilon$, $\sigma > 0.01$ will result in relatively huge bias compared with the original distribution. This illustrates that robust pattern mining can be achieved with high error-perturbed samples.

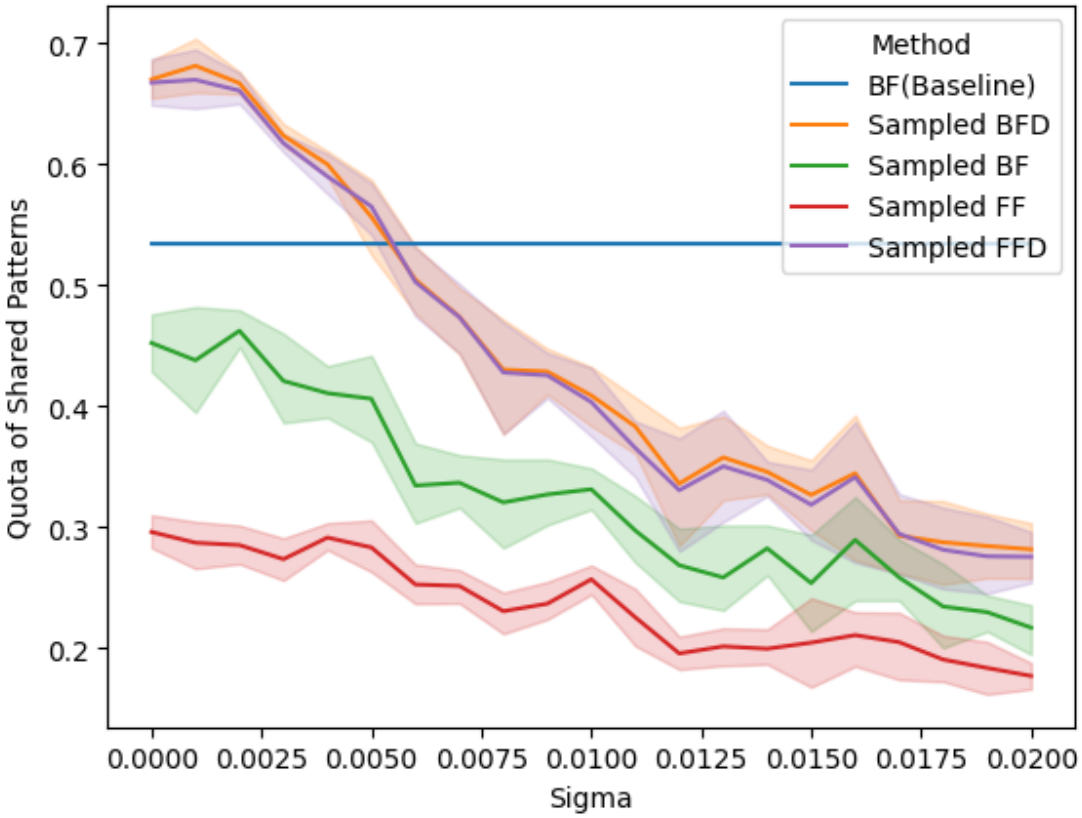


Figure 4.3: Common Pattern Quota with Increasing Error

4.3.2 Guiding Online Packing with Uncertainty

Within the OBPP, demand prediction errors remain inevitable regardless of whether unbiased estimates of the true distribution are obtainable. Consequently, the impact of demand uncertainty must be explicitly addressed during online packing processes. This section systematically analyses how such uncertainty influences the objective function value, specifically the total number of bins utilised.

The effect of uncertainty can be categorised as underestimation and overestimation. Underestimation arises when the realised quantity of a specific item type exceeds the planned allocation, resulting in surplus items that cannot be accommodated under the original packing strategy. These residual items are hereafter termed *out-of-plan* items. Conversely, overestimation occurs when the actual demand for an item type falls short of predictions, causing bins to remain unnecessarily open in anticipation of non-arriving items.

To rigorously evaluate the effects of underestimation and overestimation on bin usage, a Naive Plan Execution algorithm is employed for simulation. As detailed in Algorithm 1, this baseline strategy initially executes the precomputed packing plan. When out-of-plan items emerge, the algorithm implements a rudimentary pattern reuse mechanism: items conforming to the plan are packed as scheduled, while surplus items trigger the instantiation of a new bin and the reuse of existing patterns. This approach guarantees that all out-of-plan items are allocated to bins, albeit at the potential cost of suboptimal resource utilisation.

Algorithm 1 Naive Plan Execution

Input:Packing Plan $\mathcal{P} = (P, \mathbf{z})$, Item Sequence I

for $i \in I$ **do**
  if i is not out-of-plan **then** ▷ Follow packing plan
    if i can be packed to an opened bin according to its pattern **then**
      Pack item to that matched bin
    else
      Find a pattern p^h contains item i with $z^h > 0$
      Open a new bin with pattern p^h, then pack i into the new bin
      $z^h \leftarrow z^h - 1$
    end if
  else ▷ Reuse Pattern for out-of-plan items
    if i can pack to an opened bin according to its pattern **then**
      Pack item to that matched bin
    else
      Find a pattern p^h in current pattern set P contains i
      Open a new bin with pattern p^h, pack item i to the new bin
    end if
  end if
end for

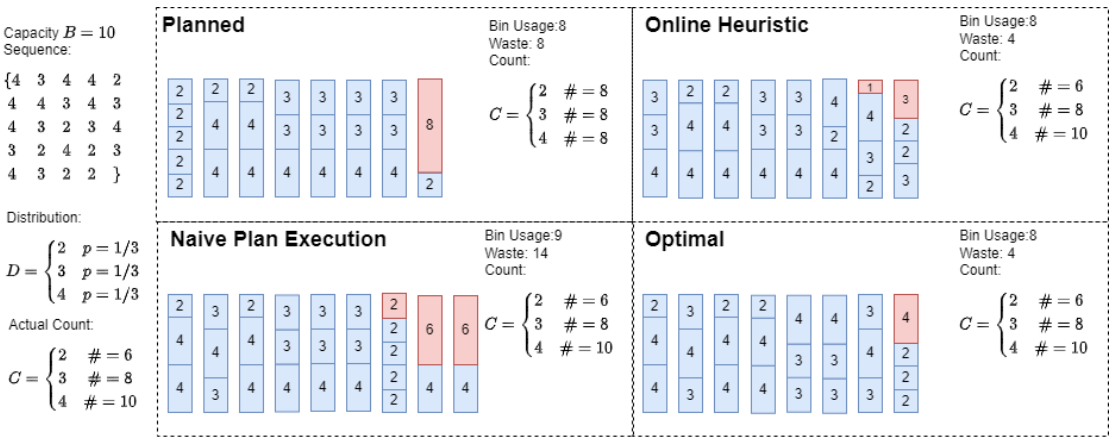


Figure 4.4: Guiding Packing by Plan with Prediction Error

Figure 4.4 illustrates the performance of distinct algorithms under erroneous demand estimation. The left panel delineates the configuration of the item sequence, while the four quadrants of the schematic compare outcomes derived from: (1) planning with imperfect demand forecasts (top-left), (2) the online algorithm BF (top-right), (3) the naive plan execution strategy (bottom-left), and (4) the optimal solution for the ground-truth sequence (bottom-right).

For the naive plan execution method, deviations from the planned demand manifest in two detrimental forms. Underestimation generates *out-of-plan* items, necessitating auxiliary bins that frequently exhibit substantial wasted capacity. Conversely, overestimation results in unutilised bin space due to anticipated items failing to materialise. A representative example occurs in the 7th bin of the naive solution: the global shortfall of size-2 items relative to forecasts leaves reserved space unoccupied, exacerbating inefficiency. It is also worth noting that the online algorithm exhibits potential suboptimal behaviour under this problem configuration. There is a size-1 waste generated by BF online heuristic, which cannot be fulfilled by possible items. As item sequences scale, such localised inefficiencies compound, progressively inflating the number of bins.

4.4 Chapter Summary

In this chapter, the plan-and-pack framework is presented, which entails initially generating patterns and corresponding quotas through estimating future item demand and solving the associated problems; this is referred to as the packing plan. The generated packing plan then guides the online packing process. During the planning stage, solutions are primarily obtained by utilising exact algorithms or heuristics as defined by the integer programme in Equations (4.2) to (4.4), which are subsequently converted into a packing plan. During the packing process, considering the unavoidable errors in demand prediction, a dynamic decision must be

made on whether to strictly adhere to the packing plan in order to achieve a more stable performance.

Section 4.3 delved into a detailed discussion regarding the regularities of pattern mining by the algorithm in a stochastic environment and the performance degradation caused by prediction error. These analyses lead to the following conclusions: Firstly, with lower prediction errors, by effectively solving an approximate problem, one can obtain patterns and distributions that are close to those derived from the offline solution of the actual problem; moreover, even with high errors, a relatively stable pattern distribution can be achieved. This provides empirical support for the performance of pattern-based algorithms. However, during the planning stage, there still remains a need for robust pattern mining methods. Secondly, in the face of inevitable prediction errors, the algorithm must seek appropriate strategies to mitigate the impact caused by such errors.

This chapter has not covered the specifics of the plan and packing implementations. These will be elaborated in the remaining chapters, including explaining how they are executed and how to address the two key issues identified in the analysis: 1) how to find effective pattern mining methods, and 2) how to minimise the impact of prediction errors.

Chapter 5

Pattern-Based Learning and Optimisation through Pricing

5.1 Introduction

For many COPs with uncertainties, deriving good patterns that are effective across different scenarios is challenging. Good patterns for one scenario may be poor for another, even if the problem structure remains unchanged. The underlying cause is that the interdependencies between the decision variables may change significantly when uncertainty is involved in the problem configuration, resulting in a decrease in performance. Therefore, it is critical to find a method to quantify the value of patterns under different problem-solving conditions accurately so that the most suitable patterns can be derived adaptively for different stochastic scenarios.

To address the aforementioned challenges, a novel scheme is proposed that can systematically generate high-value patterns for each perceived stochastic scenario and then optimise their reuse in a near-optimal manner. The proposed method in this chapter is based on the concepts of duality and shadow prices in linear programming. The effectiveness of the proposed method is examined in the 1D

OBPP, which is one of the most intensively studied COPs with many practical applications. The shadow price depicts the marginal impact of the constraints on the objective function. In the 1D BPP, this reflects the change in the optimal solution when the number of certain items in the sequence changes. By calculating the shadow prices, one can dynamically determine the importance of items for different distributions. Guided by the shadow prices, patterns with the potential to improve the objectives are generated repeatedly. This process is also referred to as column generation. For online problems, the optimal pattern combination is generated using the above method based on the latest forecast. The pattern combination is used as a packing plan to guide the online packing procedure. Owing to inherent uncertainty and imprecise forecasts, the online packing procedure must dynamically adjust the packing plan by tracking the uncertainty during packing. The proposed framework for solving the online BPP is named Column Generation Plan-and-Pack (CGPP), which will be described in detail in Section 5.2.

5.2 Mining Patterns and Guide Online Packing with Pricing

For online BPPs, the constraints have as much impact on solving the optimisation problem as the optimisation objective, which is often overlooked by most existing methods. A general framework Column Generation Plan-and-Pack (CGPP)) is proposed that explicitly adopts the dualism of COPs to assist in pattern-based solution building. For this purpose, the reformulated BPP Eq. (4.2)-(4.4) is applied. This chapter also describes the key steps and modules in the CGPP framework, including mechanisms to handle uncertainty and imperfect distribution predictions. The proposed CGPP framework is shown in Figure 5.1.

The algorithm comprises three main stages: distribution estimation, plan genera-

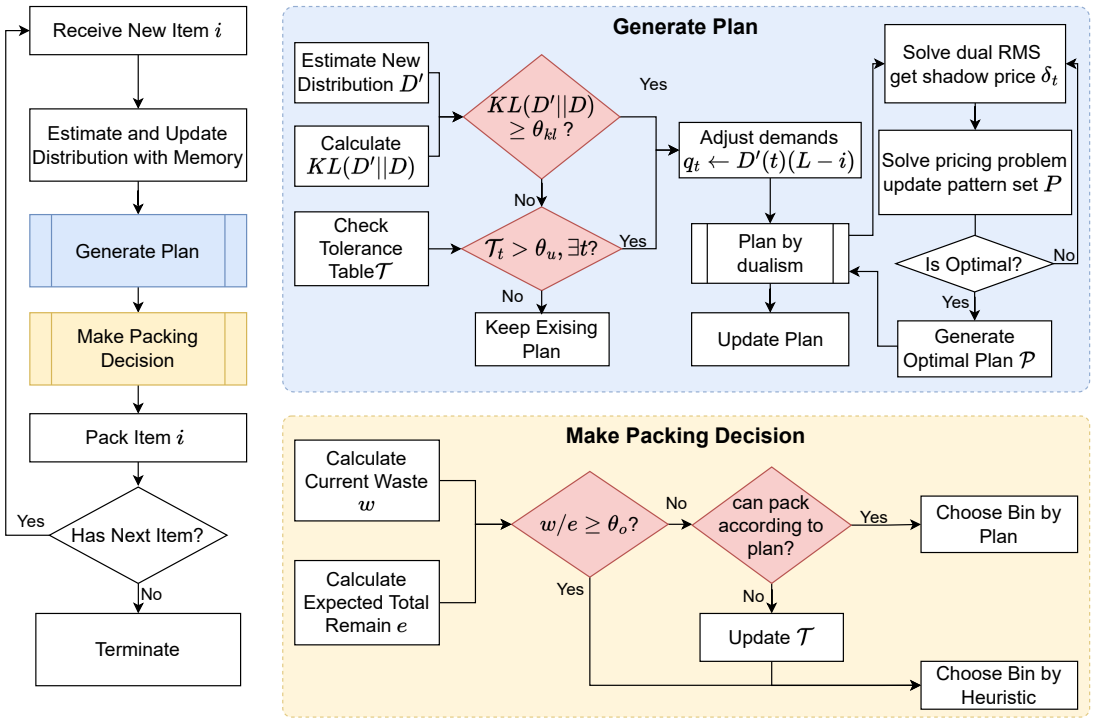


Figure 5.1: General framework of CGPP. Left: main procedure loop of iteratively packing items. Right: two critical modules of the algorithm. Red rhombuses: uncertainty handling mechanism.

tion, and packing. The distribution estimation module uses a short-term sequence memory to estimate the real-time distribution of random variables if the information is not known. The blue rectangle represents the plan generation procedure, which applies the dualism pricing method to identify good patterns and generate a plan to guide future packing. The yellow rectangle describes the adaptive packing process in CGPP under the guidance of the packing plan and on-packing heuristics in the event of poor estimation or errors. Details of this process are described in the following few subsections.

5.2.1 Planning with Shadow Pricing

As stated previously, obtaining a full $\mathcal{P}$ is not possible in most cases. Instead, the optimisation starts from a Restricted Master Problem (RMP) formulated on a subset $\mathcal{P}_r \subset \mathcal{P}$, which guarantees a feasible solution but not quality. A trivial

method for the initial $\mathcal{P}_r$ is to define a set of patterns in which each pattern packs only one type of item. In the subsequent steps, the algorithm repeatedly generates new high-value patterns to be added to $\mathcal{P}_r$ and solves the updated RMP until no new pattern is found to improve the solution further. By restricting the problem to use a small set of patterns, a much smaller RMP is considered, and only high-value patterns are added to the problem, thereby reducing the computational time considerably.

To identify high-value patterns, first the shadow price [Kuosmanen and Zhou, 2021] $\boldsymbol{\delta} = (\delta_1, \delta_2, .., \delta_T)$ is obtained for each constraint in Eq. (4.3) through the dualism property. This is achieved by solving a Lagrange dual problem of the original problem according to the dualism theory [Bonnans, 2019]. The dual problem can be written as follows:

$$\max \sum_{t=1}^{T} q_t \delta_t \tag{5.1}$$

$$\text{s.t.} \sum_{\mathbf{p}^h \in P} p_t^h \delta_t \leq 1 \tag{5.2}$$

$$\delta_t \geq 0 \qquad \forall t = 1, .., T \tag{5.3}$$

where δ_t are non-negative real decision variables. Specifically, δ_t is the Lagrange multiplier of the demand constraint with type t [Desrosiers and Lübbecke, 2005]. This can be further interpreted as how much the objective function will change if the associated demand constraint is relaxed by 1 unit.

Then, the following subproblem (known as the pricing problem or pattern gener-

ation) is solved:

$$\min\ 1 - \sum_{t=1}^{T} \delta_t p_t^* \tag{5.4}$$

$$s.t.\ \sum_{t=1}^{T} s_t p_t^* \leq B \tag{5.5}$$

and the resulting solution $\mathbf{p}^* = (p_1^*, p_2^*, ..., p_T^*)$ defines the new pattern to be added to $\mathcal{P}_r$. Eq. (5.5) is the packing constraint. The pattern generation process stops when the objective value of Eq. (5.4) becomes non-negative, which indicates that all potential cost-reducing patterns have been successfully discovered, and the resulting solution of the RMP becomes optimal. The problem (5.4)–(5.5) is a knapsack problem that can be solved efficiently when the number of item types is not large, which is the case for most real-world applications. In Algorithm 2, lines 6–10 describe the pattern generation process.

In addition to the pattern set, an important component of the RMP is forecasting the demand for each item type. The entire item sequence is divided into non-overlapping, equal-length subsequences (or sections), each of which is used to estimate the distribution of item types. Denoting the section length as L, a memory window of size $m \leq L$ is maintained. In packing step i, the distribution D' is estimated by observing items from $i-m$ to i. The KDE is utilised to determine the proportions of item types. This technique learns an appropriate linear combination of several Gaussian distributions, all sharing the same standard deviation but differing in their means. The model parameters are trained incrementally during the packing process, allowing the distribution estimation to adapt and improve over time. The demand of item type t, q_t, is set to the expected quantity of the type left in the remainder of the item sequence, that is, $q_t = D(t) * (L - i)$.

Algorithm 2 Planning through pricing by dualism at item i

Input: Memory size m, Previous plan Pl, Prior distribution D, Section size L
Parameters: Distribution threshold θ_{kl}, Underestimation tolerance θ_u

Estimate the current distribution D' with item $i-m, ..., i$
if $KL(D'||D) \geq \theta_{kl}$ or θ_u not violated **then** ▷ Determine whether the distributtion is changed
    $D \leftarrow D'$
    Estimate remaining demands $q_t \leftarrow D(t)(L-i), t = 1, 2, ..., T$
    Initialise pattern set $\mathcal{P}_r$
    while $1 - \sum_{t=1}^{T} \delta_t p_t^* < 0$ **do** ▷ Column Generation Steps
        Solve dual problem of RMP, obtain shadow prices δ_t
        Solve (5.4)-(5.5) with $\boldsymbol{\delta}$, obtain $\mathbf{p}^*$
        Update pattern set $\mathcal{P}_r \leftarrow \mathcal{P}_r \cup \{\mathbf{p}^*\}$
    end while
    Solve integer programming model Eq.(4.2)-(4.4) to obtain updated plan Pl' ▷ Obtain Packing Plan
    $Pl \leftarrow Pl'$
end if

5.2.2 Plan-Based Packing

In an ideal world, the plan generated by Algorithm 2 is implemented precisely. However, because of forecast errors in demands, additional work is required during the actual packing (see Algorithm 3). For a given packing plan $\mathcal{P}$, each newly opened bin is assigned a pattern from the plan, which implicitly specifies the type and quantity of items that should be packed. Only the items that match the assigned pattern can be packed into the corresponding bin.

Upon arrival of an item of type t, the algorithm first packs it into a matched open bin using procedure `pack_item`. If no opened bin matches the considered item, a new bin is opened and an arbitrarily feasible pattern in the current plan Pl is assigned to it. The considered item is then packed into this newly opened bin. This is performed using the procedure `open_bin_with_pattern`. When a pattern in the plan is assigned to a bin, its usage frequency must be updated accordingly. Obviously, when executing the plan, the count of the matched pattern used should not exceed the planned quota z^h in the plan.

When the demand of an item is underestimated, no feasible pattern is available

Algorithm 3 Pattern-based packing strategy at item i

Input: Packing plan Pl, Item distribution D, Opened bins $b_1, b_2, ..$, Section length L

Parameters: Overestimation tolerance threshold θ_o

Calculate remaining size $e \leftarrow (L - i) \sum_{t=1}^{T} s_t D(t)$
Calculate total empty space of opened bins w
if $w/e \geq \theta_o$ **then** ▷ On-packing uncertainty handling
    `fallback_pack`(i)
else
    if There exists an open bin b whose pattern matches i **then**
        `pack_item`(i, b)
    else if Pattern **p** in the plan matches i **then**
        $b \leftarrow$ `open_bin_with_pattern`$(\mathbf{p})$
        `pack_item`(i, b)
    else
        Update count of out-of-plan items ▷ On-packing uncertainty handling
        `fallback_pack`(i)
    end if
end if

in the plan to pack this item. In such a case, the item is packed using a fallback heuristic, for example, BF in this research. The fallback heuristic is executed using the procedure `fallback_pack` in the algorithm. The fallback heuristic either assigns an item to an existing bin, which inevitably breaks its pattern requirements, or opens a new bin to pack the item.

5.2.3 Uncertainty Handling

One critical issue for algorithms that involve prediction is dealing with the inconsistency between the prediction and reality, which is mainly caused by the natural uncertainty inherent in the problem. Prediction errors can lead to extremely poor solutions in the worst case [Angelopoulos et al., 2024]. Moreover, in real-world applications, the existence of a distribution or drift of concepts [Yu et al., 2023] can reduce the accuracy of prediction. Therefore, it is essential to design mechanisms that can ensure robust performance in error-prone predictions.

To ensure that the prediction of demands under a dynamic distribution is not far

from the real distribution, CGPP updates the distribution estimation and plan when the bias between the observed distribution D' and estimated distribution D becomes large. The Kullback-Leibler (KL) divergence $KL(D'||D)$ is applied to measure the distributional bias, using a threshold θ_{kl} to determine whether the plan should be regenerated.

To address underestimation, the out-of-plan item is allowed to have at most θ_u during packing, where θ_u represents the underestimation tolerance level. When the number of out-of-plan items does not exceed the tolerance level, they will be packed using fallback heuristics. Otherwise, the plan is considered unsuitable for the current situation. In such cases, the item distribution is re-estimated and the packing plan is regenerated accordingly. However, detecting overestimation is challenging until the very end of the item sequence because the exact item counts for each type are not accessible for the online problem. Let the risk ratio be w/e, where e is the expected sum of the remaining items for all types in the current section. The higher the risk ratio is, the more possible space is wasted eventually. A threshold θ_o is introduced to control overestimation during packing. Once the threshold is violated, the remaining items in the section will be packed using fallback heuristics. Since the decision is made at packing time, such a strategy is named on-packing uncertainty handling.

5.3 Experiments

The proposed method CGPP is tested on a wide range of online BPP datasets with different characteristics in order to establish comprehensive evaluations and understand the strengths and weaknesses of our method under different uncertainty conditions. Specifically, four distinct problem types were tested, and the details are provided in Sections 5.3.2–5.3.5. Most experiments were established with 20 instances, each having 20,000 items. Without explicitly stated otherwise,

the bin capacity was set to 100, and the item sizes were in the range $[1, 100)$.

CGPP is compared with BF, which is one of the most commonly used online heuristics due to its robustness across different scenarios and low competitive ratio (1.7), as well as three other State-of-the-art methods for online BPPs, namely ORL [Balaji et al., 2019], ProP (or ProfP for brevity) [Angelopoulos et al., 2022], and PtnP (or PatnP for brevity) [Lin et al., 2022]. An additional comparison with PtnP's updated version FPP [Lin et al., 2024] is presented in Subsection 5.3.5.

5.3.1 Algorithm Configuration

The discrepancy between the classic L2 lower bound [Martello and Toth, 1990] and the objective values obtained from various algorithms is applied to assess their performance. This lower bound has been demonstrated to be less than 1% from the optimal value. Unless otherwise specified, CGPP in this study was configured as follows: The fallback strategy was set as the one-step BF heuristic. The section length was set to $L = 1000$ with a memory length of $k = 250$ based on some initial trials. For the threshold parameters, the KL divergence threshold $\theta_{kl} = 0.1$, underestimation tolerance threshold $\theta_u = 5$, and the overestimation threshold $\theta_o = 0.8$.

The hybrid ProfilePack (ProP) algorithm was set up with the parameter $\lambda_{PP} = 0.5$ as suggested by Angelopoulos et al. [2022], because a low-error profile was not assumed in our experiment. In contrast, PatternPack (PtnP) was configured with the same parameters as those reported in Lin et al. [2022]. Both ProfilePack and PatternPack employ a statistical learning approach to learn the problem distribution dynamically. This approach entails maintaining a sliding memory window in which the item frequencies within the window are utilised to estimate the probabilities. For both algorithms, the length of the memory window was $k_{PrP} = k_{PaP} = 500$, which is the same as that reported in the literature.

The Reinforcement Learning Benchmarks for Online Stochastic Optimisation Problems (ORL) method in Balaji et al. [2019] was reimplemented using the same reported settings. This algorithm used the standard PPO algorithm [Schulman et al., 2017], with a three-layer policy network and a hidden layer of 256 nodes. The model was trained on a uniform distribution set, in which the items and bin capacity were the same as in the problem definition, unless otherwise specified. It underwent 500 epochs of training, which took approximately 600 minutes to complete on our machine.

5.3.2 Experiment Group 1:Different Items' Distributions

To evaluate the performance differences across different distributions using all algorithms, eight datasets were set up with uniform or normal distributions as the bases. These were named Uniform, Normal, Uniform-B, Uniform-C, Uniform-D, Normal-B, Normal-S, and Normal-C. Among these, six are derived datasets with the same distribution but different item range configurations. The suffix B refers to biased distribution, with item size in the range $[10, 60)$, while suffix S refers to a symmetric distribution, with item size in the range $[25, 75)$. Specifically, for Normal-B, the mean of the distribution was set to $\mu = 35$, with the same range as that for Uniform-B. The experiment with suffix C refers to the coarse experiment, with item sizes from the set $\{10, 20, ..., 90\}$.

Table 5.1 lists the results for this experiment set. It can be observed that CGPP outperformed the other methods in most experiments, except for two symmetrically distributed sets (Normal and Normal-S), for which BF outperformed all other methods. The proposed CGPP method performed particularly well for uniform distribution instances. The performance gain compared to the second-best method for these instances ranged from 17% to 62%.

Distribution	BF	ORL	ProP	PtnP	CGPP
1. Uniform	343.60	700.35	379.4	343.10	**271.75**
2. Uniform-B	199.8	339.6	546.35	249.05	**76.60**
3. Uniform-S	100.55	252.55	323.45	159.85	**79.60**
4. Uniform-C	48.15	1186.9	1010.5	562.35	**39.85**
5. Normal	**490.9**	1591.5	1722.45	2319.4	507.55
6. Normal-B	1012.65	1202.95	1165.85	1012.65	**954.70**
7. Normal-S	**77.3**	1280.55	1205.60	1738.60	167.2
8. Normal-C	490.95	494.1	2290.40	494.0	**489.65**
Overall average	345.5	881.1	1080.5	859.9	**323.4**

Table 5.1: Average objective gaps with L2 bound by different algorithms for problems with different uniform and normal distributions. Bold text represents the best average results.

5.3.3 Experiment Group 2: Packing with Prior Knowledge

This experimental set was built to investigate whether good prior knowledge of the distribution can contribute to finding a good solution. Among the algorithms discussed, BF does not rely on any learning mechanism, whereas ProP and PtnP apply a statistical approach to obtain distribution information without any prior knowledge. In contrast, ORL can be viewed as implicitly encoding the distribution information by choosing the training and testing datasets. Specifically, ORL was trained using the same distribution as that of the test datasets. In addition, the results of CGPP are reported with the exact distribution provided as prior knowledge, which is referred to as CGPP-L.

To establish a convincing comparison with ORL, three distributions are applied: BW1, LW1, and PP1, as proposed by Balaji et al. [2019]. These distributions have expected waste of $\Theta(1)$, $\Theta(\sqrt{n})$, and $\Theta(n)$, respectively. Six datasets were created (see Table 5.2). In the first three datasets (BW1-9, LW1-9, PP1-9), the bin capacity was set to 9, whereas in datasets BW1-100, LW1-100, and PP1-100, it was set to 100, following the configuration of Balaji et al. [2019]. The number of experiment instances and the associated number of items remained the same as those described in Section 5.3.2.

Distribution	BF	ORL	ProP	PtnP	CGPP	CGPP-L
9. BW1-9	**0.00**	**0.00**	527.55	**0.00**	0.25	**0.00**
10. LW1-9	103.60	156.50	388.70	103.60	223.55	**101.45**
11. PP1-9	154.90	472.75	1187.25	154.90	146.30	**145.40**
12. BW1-100	**14.10**	**14.10**	428.10	**14.10**	**14.10**	**14.10**
13. LW1-100	**0.00**	**0.00**	792.95	**0.00**	**0.00**	**0.00**
14. PP1-100	**0.00**	**0.00**	702.15	**0.00**	**0.00**	**0.00**

Table 5.2: Experimental results on distributions proposed by Balaji et al. [2019]. Bold text represents the results with the best average bin gap.

Table 5.2 presents the experimental results. Specifically, algorithms achieve an optimal solution when the result is zero, indicating no gap between the solution and the L2 lower bound. For datasets 9, 12, 13, and 14, nearly all methods attained the optimal solution, except for ProP. For datasets 10 and 11, CGPP-L outperformed all methods. However, CGPP failed to obtain competitive results for dataset 10, indicating a potential weakness of the proposed method and the importance of utilising prior knowledge, if available. It appears that the online distribution learning mechanism of CGPP misled the packing because its prior-knowledge version performed the best. ORL exhibits mixed performance: it performed badly on dataset 11 even after it was trained on the same distribution. However, PtnP achieved a good online learning strategy for this group of datasets as the result was almost the same as that of BF, whereas ProP achieved the worst performance. Overall, this group of datasets appears to be rather friendly in terms of BF, which does not involve learning.

5.3.4 Experiment Group 3: More Complex Distributions

In this section, the performance of the proposed algorithm is accessed on more complex distributions. Two groups of datasets were used. The first adopted the dual-normal distributions suggested by Burke et al. [2010], which is a typical mixed distribution. The datasets include both single- and dual-normal distributions. Our focus was on the dual part, specifically Burke 4-11, through experiments 15-22.

Distribution	BF	ORL	ProP	PtnP	CGPP
15. Burke-4	205.00	219.60	2260.0	178.65	**115.75**
16. Burke-5	167.50	179.55	202.05	165.25	**82.35**
17. Burke-6	75.20	85.20	196.85	115.45	**53.50**
18. Burke-7	50.60	56.90	120.50	78.90	**37.70**
19. Burke-8	180.55	209.20	198.4	172.00	**102.9**
20. Burke-9	145.55	157.25	165.6	140.25	**80.25**
21. Burke-10	96.55	104.20	215.7	98.10	**57.95**
22. Burke-11	54.55	61.05	185.15	73.25	**42.75**
23. Binomial-PS	**1430.5**	1703.3	2349.1	1712.0	1437.3
24. Binomial-PB	1310.9	1319.2	1551.7	1437.7	**1302.6**
25. Poisson	202.9	235.5	738.0	204.9	**177.9**

Table 5.3: Experimental results for dual and periodic distributions measured by average bin gap with the L2 bound. Bold text represents the best objective values.

Each experiment consisted of 20 instances, with each instance containing 5000 items.

The second group of experiments investigated the performance when the distribution changes periodically. All three experiments shared the same item size ranges and bin capacity configurations. The entire item sequence was divided into several equally sized sections, and each section was sampled from an independent distribution. Two groups of binomial distributions are utilised for the periodic experiments: Binomial-PS, which samples from a binomial distribution with $p = \{0.2, 0.35, ..., 0.7\}$, and Binomial-PB, which samples from a binomial distribution with $p = \{0.2, 0.3, ..., 0.6\}$. In addition, a Poisson distribution group is included with its parameters varying in the set $\{5, 15, ..., 45\}$. For each instance, the section size was set to 2000, resulting in 10 sections in total.

Table 5.3 lists the results of the two types of experiments. For the dual distribution set, CGPP significantly outperformed the other methods. Compared with BF, the reduction in the gap to L2 ranges from 21.5% to 50.1%. Compared with PtnP, the reduction was between 35.2% and 52.2%.

In the periodic distribution experiments, CGPP performed similarly to BF for the two Binomial distributions but gained a clear advantage for the Poisson distribu-

Distribution	BF	ORL	ProP	PtnP	FPP	CGPP
26. $sh = 0.5$	**0.2**	736.2	11695.8	**0.2**	154.2	476.2
27. $sh = 1.0$	153.8	1541.4	2110.2	134.4	219.4	**82.2**
28. $sh = 1.5$	608.6	2386.0	1272.6	811.4	349.2	**94.6**
29. $sh = 2.0$	1039.8	3098.6	906.8	1316.8	477.6	**133.8**
30. $sh = 5.0$	2150.6	2981.2	1641.6	2515.4	892.4	**384.2**
31. Periodic	465.4	802.2	2306.0	609.2	259.4	**208.4**

Table 5.4: Experimental results for large-scale Weibull distributions measured by average bin gap with the L2 bound. Bold text represents the best results.

tion. Compared with ProP and PtnP, CGPP had significant advantages.

5.3.5 Experiment Group 4: Large-Scale Weibull Distribution

This section investigates the effectiveness of the Weibull distribution family, which is closely connected to bin packing applications such as VM management [Castiñeiras et al., 2012]. Five different Weibull distributions are established with shape parameters $sh = \{0.5, 1.0, 1.5, 2.0, 5.0\}$. Each experiment consisted of five instances with 10^5 items. In addition, a group of datasets with periodic Weibull distributions are generated, with the shape parameters shifting to the next one stated in the list above for every 4000 items.

Some parameters were modified in this experiment to adapt to instances with a very large number of items. For CGPP, the memory length was set to $k = 1000$, the section length was set to $k = 4000$, and the underestimation tolerance was set to $\theta_u = 20$. The overestimation tolerance was $\theta_o = 1.5$ because the sequence was sufficiently long to pack items according to the plan. The memory windows of ProP, PtnP, and FPP were set to be 4000.

Table 5.4 presents the experimental results. Again, CGPP outperformed the other methods for almost all datasets except when $sh = 0.5$. One possible explanation is that the sequence heavily involved small-size items, but the algorithm continued

to assume that large-sized items would arrive in the future. The proposed method relies on a good forecast of both the types of items and their distributions. When the uncertainty is extremely high, it is probably better to revert to more myopic methods, such as BF.

For this group of experiments, the results of the bin gap from a very recent algorithm FPP [Lin et al., 2024] is also included. As shown in Table 5.4, compared to its previous version PatternPack, FPP obtained mixed results. It performed quite well for instances with $sh = 1.5, 2.0, 5.0$ and periodic instances, obtaining the second-best results among all compared algorithms. However, it is outperformed by PtnP for instances with $sh = 0.5, 1.0$, suggesting some generalisation issues.

5.4 Discussion on Solution Quality

5.4.1 Online packed solutions

Although the performance of algorithms is primarily measured by bin usage, the filled rate of all opened bins is used to investigate the solution quality and packing process when using different methods further. The bin filled rate is defined as the percentage of the total size of the items in a bin to its capacity. Two typical solutions from the Uniform-B and Normal-B datasets in Section 5.3.2 are used for the analysis. In addition, the results of the periodic Weibull dataset (Experiment 31) are analysed to observe how different methods behaved when faced with changing distributions.

Figure 5.2 shows the filled rates of the bin index for the Uniform-B and Normal-B datasets. The bin series are arranged in the order of their opening steps. The polylines of different colours are used to illustrate the average filled rates of the bins in the solutions generated by different algorithms for the given dataset. The

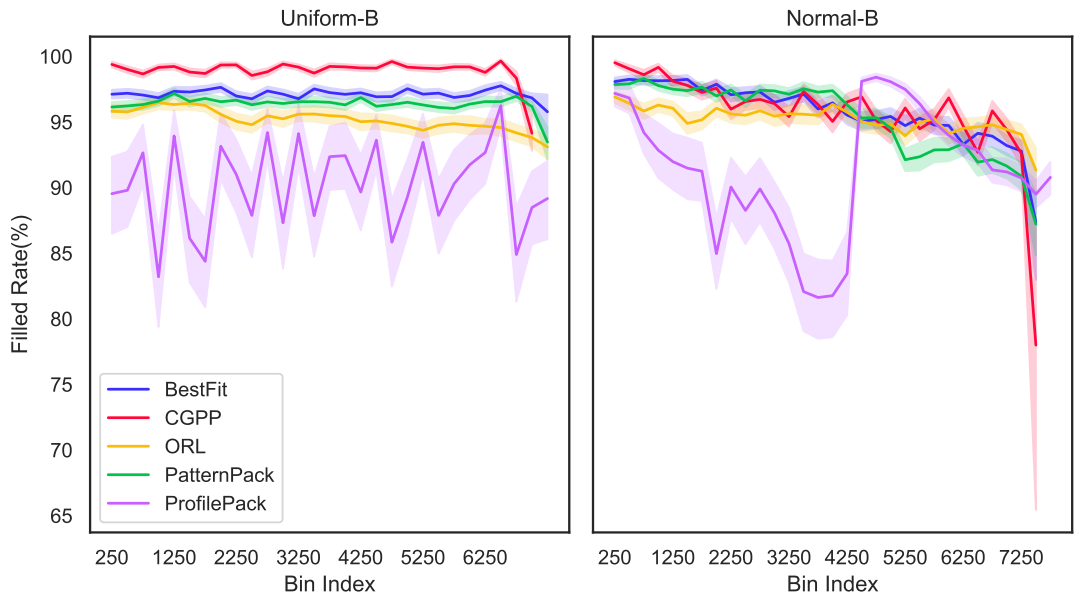


Figure 5.2: Average bin filled rates with confidence intervals.

surrounding light areas of each polyline represent the 95

Both PtnP and ORL achieved a filled rate of over 95BF achieved a slightly better filled rate than PtnP and ORL, with an average of 97.5The filled rates of ProP were consistently worse than those of the other methods. The extreme fluctuations observed for ProP illustrate the poor robustness of this algorithm. This phenomenon is likely owing to ProP lacking mechanisms for handling overestimation uncertainty, which is identified as the most wasteful resource, as discussed in Section 5.2.3.

The filled rates of BF, CGPP, and PtnP exhibited a decreasing trend on the Normal-B dataset. Clearly, the filled rate of CGPP was the highest initially, albeit with fluctuations during the entire packing stage, which could have been caused by imperfect prediction. The rapid drop in the filled rate of the final few bins by CGPP also highlights one of the main drawbacks of the CGPP method, in that overestimation is unavoidable. Similarly, PtnP and ProP also suffered from fluctuations and rapid drops owing to poor prediction. This further highlights the importance of eliminating the overestimation caused by poor predictions in

the pattern-based packing process. In contrast, ORL behaved conservatively by maintaining most bins at a similar level of filled rates for both datasets. Because ORL was trained on a uniform distribution, such a conservative strategy indicates that it cannot be generalised to other distributions.

Figure 5.3 presents the filled rates for the periodic Weibull dataset. Most methods initially achieved a filled rate of almost 100ORL still tended to sacrifice some wasted space in order to achieve a more stable filled rate. Both BF and PtnP experienced a reduction in the filled rate during the middle stage of the packing. PtnP had a slightly worse filled rate than BF and a significantly worse rate than CGPP, primarily owing to imperfect prediction when the distribution changed. ProP exhibited instability, and its performance decreased significantly when the distribution changed. At index 20000, ProfilePack switched to its fallback strategy, i.e., to pack the second-half of items with FF. However, several bins were opened with patterns that were erroneous, which could not be filled with FF, causing the fill rate to drop and therefore requiring extra bins to cover all items globally. The poor performance of ProP and PtnP highlights the potential risk of poor prediction, which misguides packing.

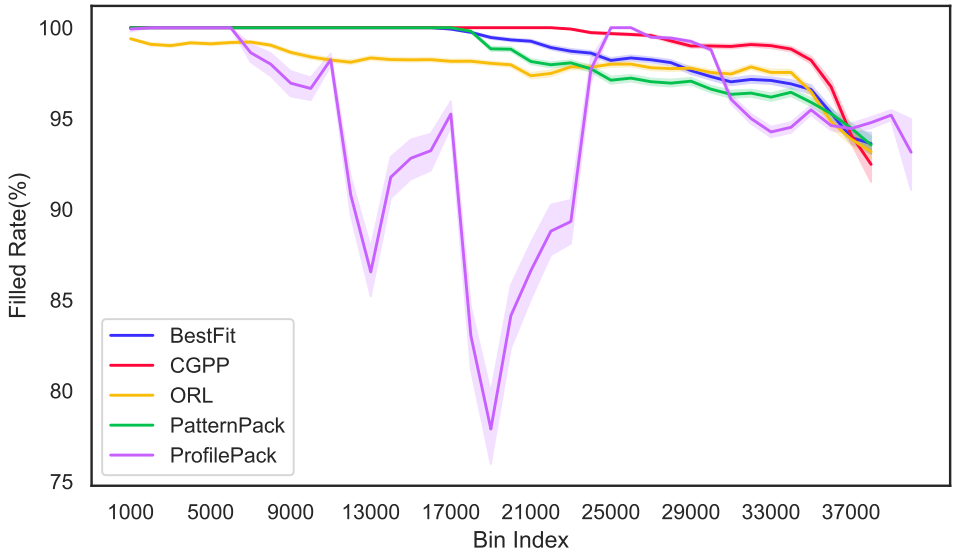


Figure 5.3: Average bin filled rates for periodic Weibull.

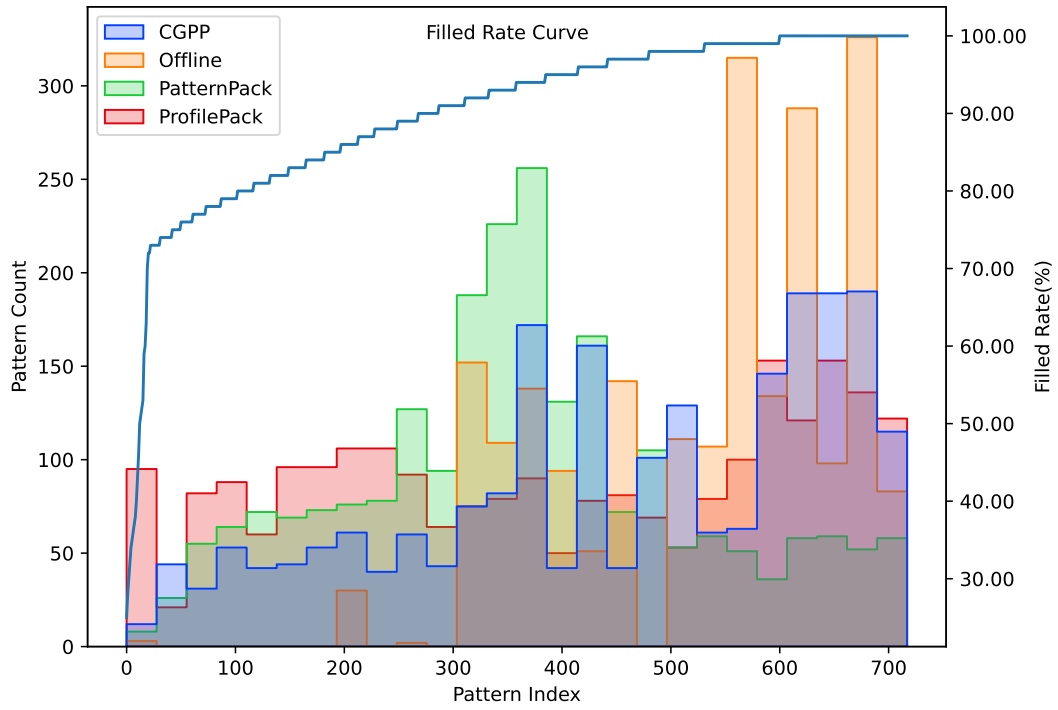


Figure 5.4: Histogram of pattern quantities and their fill rates. Blue curve: fill rate of patterns measured by the second y-axis on the right.

5.4.2 Analysis of Patterns and Their Reuse

This section analyses the pattern quality in detail and determines the extent to which the pattern contributes to achieving a good solution. An instance from Burke-4 (Experiment 15) is selected as a representative case for discussion. Similar behaviours could be observed in most other instances.

Firstly, an offline oracle solution is provided with all information known in advance. The bin patterns used in such an offline oracle solution can be regarded as high-quality patterns. It is expected that an algorithm that can recognise good patterns will tend to pack bins in a manner similar to the oracle solution. That is, not only should the high-quality patterns be used more in the online solution, but the pattern distribution should also be close to that of the offline oracle.

Figure 5.4 presents a histogram of the solution patterns. All patterns were sorted

according to their fill rates, and each pattern was assigned a unique index, with a larger index indicating a higher fill rate. The changes in the fill rates across different pattern indices are represented by the blue curve in the figure (measured by the second y-axis on the right). The height of each histogram bar represents the quantity of a certain pattern used in the solution. The Offline histogram represents the pattern distribution for the offline oracle, where the patterns are considered high quality.

In comparison, CGPP achieved a histogram that closely resembled the offline solution, with more high-quality patterns used and a much higher overlap with the oracle solution. This indicates that not only was CGPP able to identify good patterns, but it could also effectively reuse those patterns to reduce the overall waste, resulting in improved bin usage.

For PtnP, the ability to reuse patterns was also demonstrated. However, it favoured patterns at indices 300–400, resulting in not only a high frequency of suboptimal patterns (90%–95% fill rate), but also low overlap with oracle patterns. For ProP, the patterns were more evenly distributed. It achieved a better overlap at indices 500–700 than PtnP, but it also used many patterns of low fill rates (e.g. pattern indices 0–300), which rarely appeared in the offline oracle. These low-quality patterns resulted in poorer overall performance.

5.4.3 Discussion of Compared Methods

For most datasets in our experiment, CGPP outperformed BF and other existing methods. This advantage can mainly be attributed to the dynamic pattern identification, the associated planning, and subsequently, the reactive fallback strategy. With the distribution of random variables being provided, CGPP could achieve near-optimal solutions, significantly outperforming all other methods. Under unknown distributions, it achieved excellent performance in most cases compared to

the other methods.

The proposed method significantly outperformed the ORL approach. ORL relies strongly on the distribution during training and fails to generalise to out-of-distribution data. In contrast, CGPP performs adaptive online learning; thus, it is less dependent on the original estimate of the distribution.

Our method exhibited superior performance in terms of average bin usage compared to PtnP and its fuzzy-enhanced version FPP, as it utilises dualism pricing-based column generation for planning. This yielded better patterns than the online heuristics employed by PtnP.

ProP theoretically proved that applying good prediction can lead to high-quality solutions. However, it lacks an uncertainty handling strategy in terms of implementation.

In special cases (e.g. Experiment 26), ProP generated extremely poor results, indicating its major reliability issues. However, the success of CGPP in unknown distributions indicates the importance of handling uncertainty appropriately.

5.5 Further Analysis of CGPP

5.5.1 Sensitivity Analysis

To determine how the parameters influence the performance, this section conducted sensitivity analyses of CGPP with normal, uniform, and Weibull distributions with $sh = 2.0$ and 5.0, each using 4000 items. The basic configuration was the same as that described in Section 5.3.1 and only one parameter was altered for each experiment. Figure 5.5 shows the performance gap of the method with varying parameters: a memory length of $[50, 1000]$, KL threshold of $[0.01, 1.0]$,

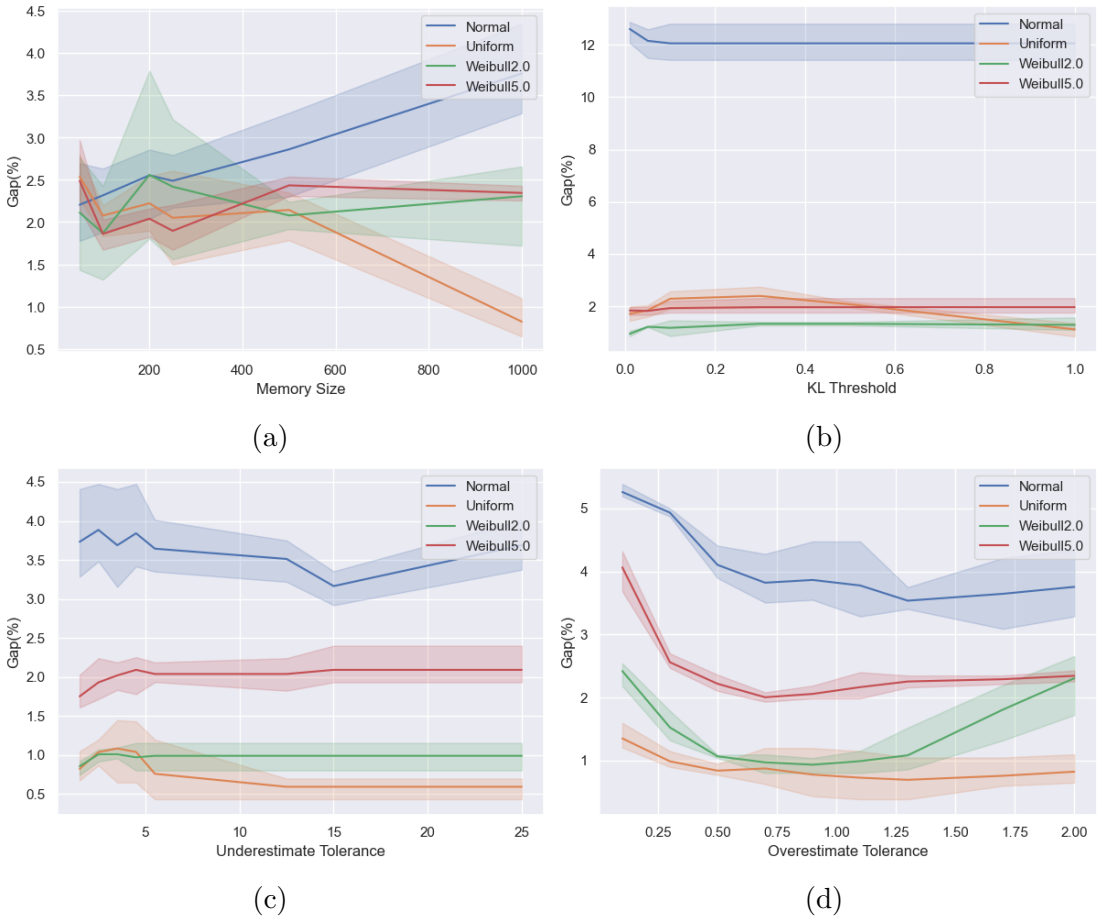


Figure 5.5: Gap curve with different parameter sets of CGPP: (a) memory length, (b) KL threshold, (c) underestimation tolerance, and (d) overestimation tolerance.

underestimation tolerance of $[3, 25]$, and overestimation tolerance of $[0.1, 2.0]$.

Some observations can be made from the experiment. First, increasing the memory length leads to better estimation, but a large window may reduce the sensitivity level of the detection of the distribution drifts, thereby reducing the performance. The KL threshold is associated with the sensitivity of distribution detection and remains stable in the proposed algorithm. Second, the underestimation tolerance determines the frequency of the plan to be fine-tuned without re-estimation. Simple distributions such as uniform and normal distributions require a less frequent fine-tuning strategy, whereas complex distributions benefit from more frequent updates. Finally, the overestimation threshold controls when the fallback heuristic will intervene in the packing decision. As shown in Figure 5.5d, a balance between the plan and fallback was achieved in the range of $[0.7, 1.3]$. In general,

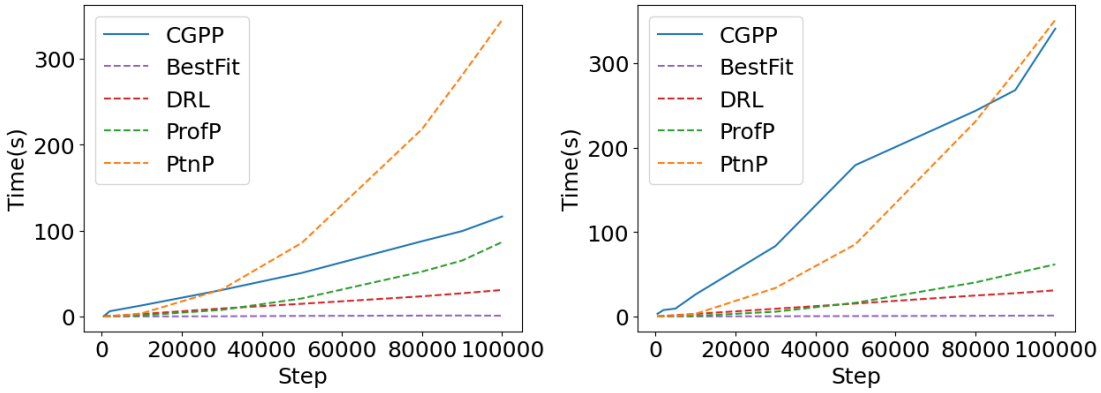
the memory length and overestimation threshold have the strongest influence on performance, while the KL threshold and underestimation threshold play a more supportive role.

In conclusion, the algorithm is robust overall, with the largest performance fluctuation observed being a 2% gap. However, according to the no-free-lunch theorem, no single parameter set achieves optimal performance across all possible distributions. Therefore, these parameters must be adjusted specifically for practical purposes.

5.5.2 Time Cost Analysis

CGPP consists of column-generation-based planning and rule-based packing. If parallel computing is not involved, the time complexity of CGPP is dominated by the planning part. Column generation for planning can converge quickly with proper stabilisation methods [Lübbecke and Desrosiers, 2005], which means that pattern generation will be executed in finite steps. The pattern generation iteration, although NP-hard, can be solved efficiently in practice. A recent study [Jin, 2024] on a pseudo-polynomial time algorithm for the knapsack problem shows that fast algorithms exist with respect to item type T and the range of the shadow price δ, both of which are bounded in our case. In the packing part, Algorithm 3 involves packing by patterns and calling the fallback heuristics. The packing stage requires searching the pattern table and current open bins, requiring at most $O(N/T)$ for each packing step with the proper data structures. Therefore, the total packing time complexity for a fixed plan is $O(N^2/T)$. In comparison, packing using the BF fallback heuristic costs a maximum of $O(N)$.

The total time cost curve in Figure 5.6 is presented using a Weibull distribution with a shape parameter of $sh = 2.0$ as representative of both stationary and periodic Weibull distributions. The size of the instance is increased from 500 to



(a) (b)

Figure 5.6: Total time cost of CGPP under (a) stationary distribution with $sh = 2.0$ and (b) periodic distribution.

100,000, while maintaining all other configurations the same as those described in Section 5.3.5. For stationary distributions, CGPP was executed effectively and outperformed PtnP. The periodic distribution requires more frequent planning; therefore, the time cost increases in such cases. In this scenario, CGPP had a similar time cost in the long term to PtnP.

5.5.3 Scalability and Generalisation

This chapter conducted a total of 31 experiments on different configurations, not only using different distributions but also scaling the sequence length from 20,000 to 100,000. In addition, the experiments involved problems with the number of item types ranging from 9 to 100. The experiments showed that our methods could maintain excellent performance for most of these configurations.

Although the discussion and experiments are primarily on the online 1D BPP, it is possible to claim that the framework can be applied with minimal modifications to problems with similar structures. Specifically, the framework can be migrated to problems that can be formed through a combination of reusable subsolutions. A straightforward example is the online capacitated vehicle routing problem, which can be formulated as a combination of 1D bin packing and the travelling salesman

problem. Other problems include cloud computing resource management [Sheng et al., 2022] and scheduling [Leeftink and Hans, 2018].

5.6 Chapter Summary

This chapter showed that the values of patterns could change owing to uncertainties related to objectives and constraints and that most existing methods fail to exploit the interdependencies among decision variables incurred by uncertainties in constraints. This chapter established a scheme to quantify the usefulness of different patterns dynamically based on the dualism of the COP and used the information to guide the decision process. To handle the influence of both underestimation and overestimation, threshold-based methods are introduced to eliminate the inconsistency between the plan and observation.

The test results on the BPP showed a significant performance advantage of the proposed method compared with current state-of-the-art approaches. However, there are still cases in which the proposed method is less advanced, suggesting the need for further research. Another drawback is the cost of planning, which requires heavy computation in dynamic and uncertain environments. Finally, our method is restricted to well-structured problems that can be conveniently formulated with linear/integer mathematical models and are suitable for column-generation-based approaches.

Chapter 6

Online Risk-Aware Pattern Adjustment

6.1 Introduction

When accurate distribution information is unavailable or when distribution parameters change dynamically, prediction-based algorithms often rely on online learning from historical data to forecast the item sequence for the future horizon. This approach typically results in unavoidable prediction errors, especially when the distribution is changing.

To solve dynamic problems, a common strategy is the rolling-horizon [Glomb et al., 2022]. This strategy involves continuously replanning the future based on a predefined planning cycle. It assumes that the distribution remains stable within the planning cycle and that the plan is efficient, allowing errors to be ignored. However, for OBPP, prediction errors can lead to a significant performance drop [Angelopoulos et al., 2024]. Consequently, rolling-horizon strategies require a high level of prediction accuracy, which may be unattainable in some online problems. Furthermore, when the distribution changes, the rolling horizon strategy

necessitates waiting until the end of the cycle to update the plan, often resulting in delayed responses and performance decline.

Another approach is to use a hybrid method that combines prediction-based algorithms with a zero-knowledge online algorithm to ensure the overall robustness of the algorithm. This scheme can be described as follows: first, the algorithm generates a plan for the future based on predictions that contain errors. Next, the algorithm sets a termination condition. As long as this condition is not met, the algorithm follows the plan. Once the termination condition is satisfied, the algorithm switches to a fallback strategy that does not rely on the plan. These fallback strategies are typically zero-knowledge online algorithms, such as FF.

A common strategy for constructing termination conditions involves using thresholds. One approach is to set thresholds based on the number of steps. For instance, Angelopoulos et al. [2023] counts the packed items according to the plan and, upon reaching a certain quota of total items, utilises FF for the remaining items. A similar strategy is employed by Lin et al. [2022], which updates the plan at fixed intervals. Once the plan is updated at these fixed steps, non-filled bins are marked as fallback bins, which will subsequently pack out-of-plan items that cannot be packed according to the pattern using BF. In the aforementioned CGPP (Chapter 5), the termination condition is constructed based on the expected sum of future item sizes.

These strategies share a common issue: the fallback decision usually makes decision on each packing step, therefore being myopic. Additionally, the termination conditions rely on manual design, causing the strategies to struggle to avoid response delays. If the termination condition is too strict, the algorithm behaves like a fallback zero-knowledge algorithm, losing the potential to enhance online performance through predictions. Conversely, if the condition is too lenient, erroneous predictions and plans may mislead the online packing process. Furthermore, a deeper problem introduced by hybrid methods is that the algorithm often finds

it difficult to distinguish between the effective and ineffective parts of the plan. Fallback strategies may undermine originally effective planning, leading to a decline in planning efficacy and negatively impacting the algorithm's performance over the long term.

Based on these observations, this chapter proposes the Online Risk-Aware Pattern Adjustment (ORAPA) strategy. This strategy adds an extra adjustment layer to the basic plan-and-pack framework to modify the already planned patterns. It consists of two main steps: First, the algorithm assesses which patterns may pose risks—specifically, those that may not be packable according to the current pattern—by evaluating the differences between the shadow prices derived from the distribution at planning time and the latest observed distribution.

Second, the algorithm constructs a problem to generate alternative patterns based on these two distributions and solves it. Notably, risky patterns may belong to bins that have already been partially packed, resulting in corresponding changes to bin capacity. These patterns are referred to as remnant patterns. Due to the existence of remnant patterns, simply solving the original BPP problem is no longer applicable. This issue will be modelled as Stochastic Multiple Knapsack Problem (SMKP), ensuring that the algorithm can address variable-sized bin capacity problems while also obtaining robust alternative patterns.

6.2 Related Works for SMKP

This study addresses the issue of simultaneously adjusting the patterns for already opened bins to achieve a better solution. This requires re-formulating the planning problem for online bin packing to be Stochastic Multiple Knapsack Problem (SMKP), which will be discussed later. Since there is limited literature on the exact same problem, a wider range of literature reviews is presented on similar problems, including variable-sized bin packing problems (VSBPP) and stochastic

knapsack problems (SKP). The VSBPP has been examined within the research community, with Haouari and Serairi [2009] proposing a genetic algorithm for static offline VSBPP and Crainic et al. [2011], Correia et al. [2008] discussing lower bounds and heuristics, as well as linear integer programming models. Turky et al. [2020] discussed a similar deterministic problem with multiple types of capacity constraints (e.g. weights and volume). Perboli et al. [2012], Baldi and Bruglieri [2017] explore stochastic generalised BPP, where item sizes are deterministic, but profits are stochastic. Martinovic and Selch [2021] discusses exact methods for bin packing problems with uniform bin sizes, while Yan et al. [2022] examines stochastic BPP within a cloud computing context, characterised by limited available bins for cloud requests. Crainic et al. [2016] considers the cost implications of stochastic-sized bins in logistics. Li et al. [2022], Zhao et al. [2024] apply deep reinforcement learning to multi-dimensional strip packing problems, inherently incorporating stochastic elements through DRL-based modelling.

6.3 Risky Patterns: Identification and Adjustment

6.3.1 Additional Adjustment Stage

Based on the observations above and an analysis of the shortcomings of existing methods, this section proposes ORAPA, an additional adjustment layer built upon the plan-and-pack framework. The goal of ORAPA is to adapt the existing plan to newly observed conditions through fine-tuning. Specifically, this layer will utilise shorter cycles than the planning section to capture the differences between the latest distribution and the plan-time distribution while also identifying the risks associated with patterns. It is important to note that the ORAPA stage does not alter the original planning and packing strategies. Figure 6.1 illustrates the

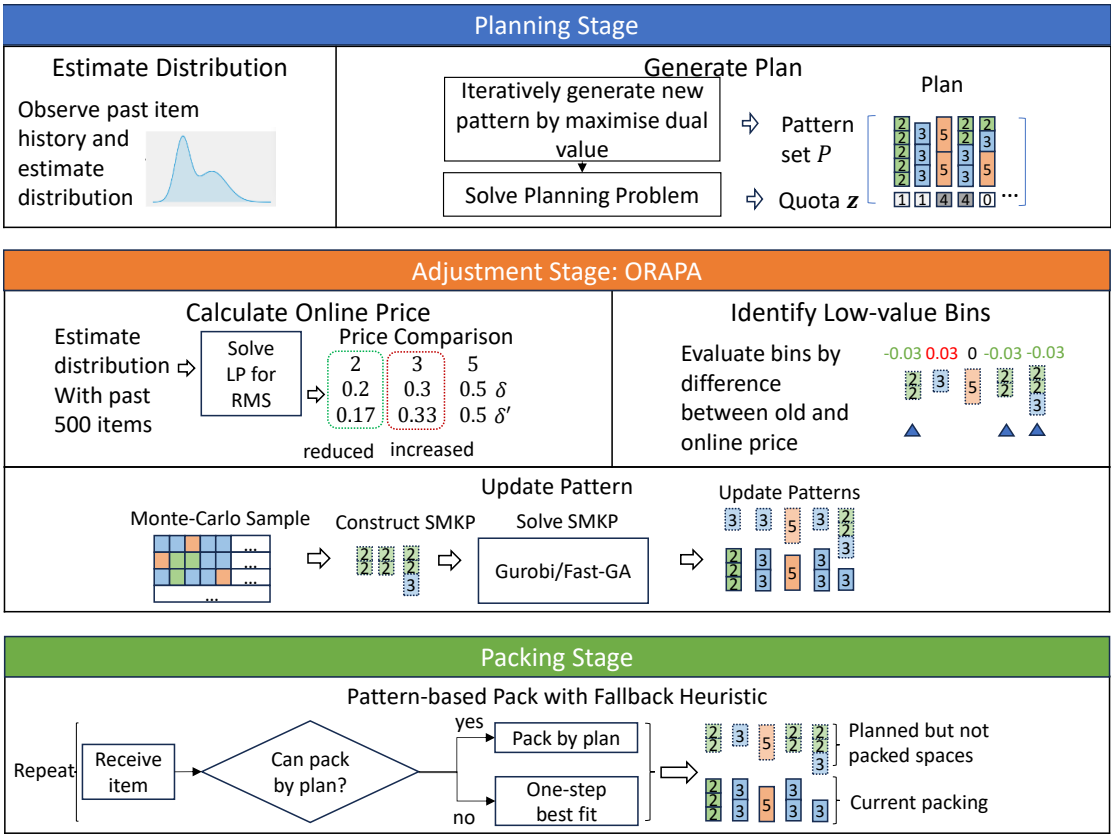


Figure 6.1: Overview of structure of proposed methods

complete framework of the proposed ORAPA stage. Algorithm 4 represents the new three-stage framework with the proposed ORAPA tactical stage. Specifically, the planning (line 3) is done by Algorithm 2, and the packing stage is done by Algorithm 3. Unlike the aforementioned hybrid methods, ORAPA does not adjust during on-packing. Instead, it brings uncertainty handling to the forefront, enabling fine-tuning of the overall plan rather than making decisions for a single bin.

Two principal challenges are encountered in dynamically updating the patterns for bins which have been initiated: determining what patterns should be altered in the existing plan and generating appropriate patterns for replacement. The benefit is aimed to be retained by having the algorithm plan for the long-term through solving the set-covering formulation while maintaining flexibility towards prediction errors. Consequently, it is desirable for the pattern updating to be limited to a minimal range of bins. Online pricing is introduced as an evaluative

Algorithm 4 Plan-and-pack Framework with ORAPA.

Input: Item Sequence I, Planned pattern set $\mathcal{P}_r$, Remnant patterns for opened bins $\mathcal{R}$, Planning stage demand $\mathbf{q}^0$, latest demand $\mathbf{q}'$

for $i \in I$ **do**
  if $i \mod L = 0$ **then**
    Plan for packing next L items.
  end if
  if $i \mod m = 0$ **then**
    Solve Eq.(5.1)-(5.3) with $\mathcal{P}_r$ and $\mathbf{q}^0$ to gain $\boldsymbol{\delta}^0$
    Solve Eq.(5.1)-(5.3) with $\mathcal{P}_r$ and $\mathbf{q}'$ to gain $\boldsymbol{\delta}'$
    $\mathcal{R}' \leftarrow \{\mathbf{r} \in \mathcal{R} | \texttt{is\_risky}(\mathbf{r}, \boldsymbol{\delta}^0, \boldsymbol{\delta}')\}$ ▷ Find risky remnant patterns with Eq.(6.1)
    $\hat{\mathcal{R}} \leftarrow \texttt{solve\_smkp}(\mathtt{R}', \boldsymbol{\delta}')$
    Replace $\mathcal{R}'$ with $\hat{\mathcal{R}}$
  end if
  Pack item i by updated patterns.
end for

tool for gauging the quality of open bins based on the most recent observations. This is achieved by frequently solving the relaxed dual problem during execution. A shorter range of historical items will be used to estimate the distribution for flexibility considerations. Consequently, bins with lower values will be pinpointed through scrutiny of the unoccupied portions of the patterns. A detailed discussion of the evaluative mechanism will be presented in Section 6.3.2.

Upon the identification of bins with lower value, the issue can be conceptualised as a Stochastic Multiple Knapsack Problem (SMKP). Specifically, the objective is to minimise the likelihood of high-value bins being overlooked by the patterns, as employing heuristics to pack a high-value item could potentially diminish the overall objective value. Therefore, items are weighted according to their values through an online pricing mechanism. The formulation of the problem is delineated in Section 6.3.3.

6.3.2 Identify Risky Bins by Online Pricing

Fine-tuning the existing plan requires accurately identifying which patterns may cause issues rather than completely overhauling the entire plan. This necessity stems from the online nature of the process, where changes to patterns need to be made as quickly as possible. Additionally, it is generally true that the plan may still hold a degree of validity in the long term, making a complete re-planning unnecessary. To address this, an online pricing mechanism is proposed to evaluate patterns, particularly the risks associated with remnant patterns that could lead to performance degradation.

The online pricing mechanism utilises a sliding window that is smaller than the planning horizon as memory. This window is employed to estimate the latest distribution and further calculate the shadow prices of the current patterns.

Specifically, the whole item sequence is divided by equal-length sections L, which is the planning horizon. Let the distribution be D^0 and the associated demand be $\mathbf{q}^0$ at the planning stage. Once the pattern subset $\mathcal{P}_r$ is determined, the plan-time shadow price can be calculated, denoted as δ^0. During the packing procedure, the initial estimation is then updated regularly in an online, rolling-horizon manner, denoted as D'.

A sliding window is applied to estimate the online distribution of length m for estimating the latest distribution D' and corresponding demand $\mathbf{q}'$. By solving the dual problem of Equation (5.1)-(5.3) with the existing pattern set, the online price $\boldsymbol{\delta}'$ can be obtained.

Since the shadow price represents the impact on the function value when the quantity of item types changes, the difference between the plan-time price and the online price indicates a shift in their importance. When an item type has a low plan-time price and a high online price, it suggests that executing according to the

current pattern set may result in an insufficient supply of that item, potentially requiring additional bins in the future to pack them. Conversely, when an item type has a high plan-time price and a low online price, it indicates a transition from insufficient to sufficient supply, signalling that overestimation may have occurred.

Once the plan-time price and online price of an item are obtained, risk assessments for the patterns can be conducted. Since the patterns that remain unallocated in the plan can be addressed in future column-generation planning, ORAPA specifically focuses on the bins that are already open and the patterns allocated to them.

For opened bins, items in the pattern can be classified into those that have arrived and those that have not. The items that have already arrived cannot be modified according to the definition of online problems, while the items that have not arrived in the pattern can influence future packing. These not arrived items are named as remnant pattern, denoted by vector $\mathbf{r}$. Therefore, the value of opened bins is calculated by the sum of prices for the remnant pattern.

The identification of high-risk bins primarily relies on the fluctuations in their value following the latest forecast. Specifically, there are two types of pricing for bins that are upheld: the plan-time price for pattern, derived from $\mathbf{r} \cdot \boldsymbol{\delta}^0$, and the online price for pattern, calculated by $\mathbf{r} \cdot \delta'$.

Similarly, when the online price is lower than the plan-time price, it indicates that the pattern is likely to contain overestimated items, which could lead to associated risks. Moreover, when both the online price and plan-time price are zero, it indicates that the items included in the pattern are sufficiently supplied in both scenarios. Although this is not as apparent as a price decrease, it may also signify the presence of items in an overestimated state within the pattern. There is an exception for the case when both prices are zero: when there is very little remaining space in the bin, only small items may fit into it. In this situation,

adjusting their patterns has minimal global impact. This leftover space often leads to remnant patterns being used to cover small items. Since these items have a negligible effect on the objective function value, it is easy for the plan-time price and online price to both be zero when calculating the shadow price, which can interfere with risk assessment. As a result, these bins are excluded from the risk analysis. The criterion for determining whether a bin is high-risk or not is encapsulated in Eq. (6.1):

$$((\mathbf{r} \cdot \boldsymbol{\delta}^0 > \mathbf{r} \cdot \boldsymbol{\delta}') \vee (\mathbf{r} \cdot \boldsymbol{\delta}^0 = \mathbf{r} \cdot \boldsymbol{\delta}' = 0)) \wedge (1 - \frac{\sum_{t=1}^{T} r_t s_t}{B} \geq 0.1) \tag{6.1}$$

6.3.3 Stochastic Multiple Knapsack Problem(SMKP) for Pattern Update

Risky remnant patterns should be replaced with patterns that are more robust towards the future. This means that in any potential future sequences, new patterns must perform well. Since the selected risky patterns and their associated bins are fixed, this implies that the updated patterns will be able to achieve a higher expected total price.

The empty space for opened bins may vary, therefore, the standard bin packing formulation cannot be applied. The pattern updating problem is formulated as a stochastic multiple knapsack problem. Consider U bins with different capacities, each has a unique index u and capacity b_u.

The formulation of SMKP is given as Equation (6.2)-(6.4):

$$\max \mathbb{E}[\sum_{t=1}^{T} \delta'_t \min(q'_t, \sum_{u=1}^{U} x_{ut})] \tag{6.2}$$

$$s.t. \sum_{t=1}^{T} x_{ut} s_t \leq b_u, \qquad \forall u = 1..U \tag{6.3}$$

$$x_{ut} \in \mathbb{N} \qquad \forall t = 1..T, u = 1..U \tag{6.4}$$

The random variable $q't$ denotes the demand for item type t in a potential incoming sequence, representing the quantity of items. The non-negative integer decision variable xut determines the quantity of item t to be included in the new pattern of the variable-sized bin u. $\delta't$ represents the online price of items. Equation (6.2) outlines the objective function, which aims to maximise the expected total price for items aligned with the pattern, i.e., items that can be packed according to the patterns. For a specific type t, the total number of pattern-aligned items across all variable-sized bins can be calculated as $\min(q't, \sum u = 1^U xut)$. Equation (6.3) represents the capacity constraint for all variable-sized bins, while Equation (6.4) signifies the non-negative integer constraint for decision variables.

Due to the use of a smaller sliding window for online pricing calculations, the corresponding distribution estimation may introduce noise. Additionally, the same distribution can generate different item sequences, leading to inconsistencies in the number of item types. Therefore, the Monte-Carlo simulation is employed to approximate the potential future sequences. Consider the incoming item sequence of length V, K new sequences are sampled with the latest forecasted online distribution D'. The demand of each sample can be calculated by counting, denoted as q'_{kt} for each type t. An efficient solution should perform well across potential samples.

Additionally, the positive shadow prices are sparse. This is due to the shadow

price indicating the importance of demand constraints. However, a zero shadow price does not mean that the type of item is not important and should be excluded from the solution. Empty patterns may be caused by focusing only on maximising the objective function. Therefore, all bins are forced to be filled. Moreover, the original problem is not linear as the objective function involves a minimising operator. The linearised problem is shown with the Monte-Carlo simulation as Eq.(6.5)-(6.9).

$$\max_{(x,y)} \frac{1}{K} \sum_{k=1}^{K} \sum_{t=1}^{T} \delta'_t y_{kt} \tag{6.5}$$

$$s.t. \sum_{t=1}^{T} x_{ut} s_t = b_u, \qquad \forall u = 1, ..., U \tag{6.6}$$

$$y_{kt} \leq q'_{kt}, \qquad \forall k = 1..K, \forall t = 1, ..., T \tag{6.7}$$

$$y_{kt} \leq \sum_{u=1}^{U} x_{ut} \qquad \forall k = 1..K, \forall t = 1, ..., T \tag{6.8}$$

$$x_{ut} \in \mathbb{N} \qquad \forall t = 1, ..., T, u = 1, ..., U \tag{6.9}$$

Slack variable $y_{kt} = \min(q'_t, \sum_{u=1}^{U} x_{ut})$ is introduced for linearise. Additional constraints of Eq. (6.7)-(6.8) are added to ensure that maximising the linearised problem is equivalent to the original problem. The capacity constraint is changed to the equation as Eq. (6.6).

6.3.4 Solve SMKP

Two algorithms have been proposed to address the Stochastic Multiple Knapsack Problem: firstly, the problem is solved using an integer programming solver; secondly, a fast evolution algorithm has been introduced to ensure promising performance within a limited computational time frame. Furthermore, a Monte Carlo sampling-enhanced A-BFD algorithm has been established as a benchmark.

The linearised Stochastic Multiple Knapsack Problem with Monte-Carlo sampling, as outlined in Eq.(6.5)-(6.9), can be efficiently solved using existing integer programming solvers. In this study, the Gurobi optimiser has been selected for addressing the integer linear programming problem. According to the latest known comparative results for integer programming solvers [Mittelmann, 2017], the Gurobi optimiser has demonstrated superior performance across all benchmarks.

Furthermore, a fast genetic algorithm (FGA) has been developed to tackle the Stochastic Multiple Knapsack Problem, detailed in Algorithm 5. The FGA is enhanced with neighbourhood search techniques to expedite convergence. The chromosome is structured as a $U \times T$ integer matrix $\mathbf{X}$, with each cell corresponding to the decision variable x_{ut}. Notably, each column in the matrix $\mathbf{X}$, indexed by u, represents the replacement pattern for bin u. The fitness value of each individual is computed using Eq. (6.5) with sampled demands $\mathbf{q}'$ and the latest price $\boldsymbol{\delta}'$. Crossover operations are conducted at the bin level to prevent exceeding capacity limits. Mutation involves exploring different neighbourhoods. Initially, an item-level neighbourhood is defined, involving slight random adjustments to individual items. Secondly, a bin-level neighbourhood is established, which entails removing items from randomly selected bins and replacing them with randomly sampled items. These neighbourhood search steps are delineated in lines 5-16. Additionally, the capacity constraint is ensured by packing the largest possible items into bins with available space.

The A-BFD algorithm, as outlined in Crainic et al. [2011], is a simple yet powerful heuristic for solving deterministic multiple knapsack or bin packing problems. This algorithm is considered a baseline for assessing the performance of solving SMKP. The algorithm initially arranges the bins and items in descending order, after which all items are packed using the Best Fit (BF) strategy until no further items can be accommodated within the bins. The original version of A-BFD was designed

Algorithm 5 Fast-GA for solving SMKP.

Input: Capacity of bins $b_1, b_2..b_u$, demand $\mathbf{q}'$, price $\boldsymbol{\delta}'$

GA parameters Parameters: Number of population N_p, generation N_g, selection quantity N_s, crossover probability p_c,mutation probability p_m,local search parameter N_{item}, N_{bin}

Initialise population $Pop = (\mathbf{X}^0, \mathbf{X}^1, ..., \mathbf{X}^{N_p})$ by A-BFD solution with randomly sampled items.
for $j = 1..N_g$ **do**
    Select μ individuals from Pop
    Execute uniform crossover with chosen individuals
    for $\mathbf{X}$ in chosen individuals **do** ▷ Mutation
        for $k = 1, ..., N_{item}$ **do** ▷ Item level local search
            Randomly select a bin b' and an item in the bin's remnant pattern i'
            Change the item with a small offset
            if Found better individual $\mathbf{X}'$ **then**
                $\mathbf{X} \leftarrow \mathbf{X}'$
            end if
        end for
        if Not found better individual **then** ▷ Bin level local search
            Destory N_{bin} bins
            $\mathbf{X} \leftarrow$ Reconstruct destoryed bins with new sampled items
        end if
    end for
    if Any bin has empty space **then** ▷ Satisfy Constraint(6.6)
        Fill bin with a largest possible item
    end if
end for
Return individual with highest fitness value in final population

to solve deterministic problems. To address the Stochastic Multiple Knapsack Problem, the Monte-Carlo simulation is used to enhance A-BFD. Specifically, several samples are drawn from the distribution D', and the deterministic A-BFD algorithm is executed for each sample to obtain different solutions. Among these solutions, the one with the maximum objective function value is selected as the final solution.

For solving the Stochastic Multiple Knapsack Problem, the number of item types determines the algorithm's time cost, given a certain number of samples. Integer programming solvers primarily optimise the search process by traversing the space of integer variables, which typically results in exponential growth in time cost

as the number of variables (i.e. item types) increases [Chen et al., 2011]. On the other hand, the FGA confines the search space to a manageable range by employing domain-specific search operators designed for the Stochastic Multiple Knapsack Problem, thus avoiding exponential growth in time cost. However, the FGA suffers from a common issue among metaheuristic algorithms, namely the inability to guarantee convergence. In scenarios with relatively few item types, the FGA often produces suboptimal solutions. For the A-BFD algorithm, its time cost additionally depends on the length of the sampled instances due to its reliance on sorting and the Best Fit strategy. With proper implementation, for a single instance, A-BFD will achieve a time complexity of $O(n \log n)$, where n denotes the length of the sampled instance. As will be seen in the next section, under typical experimental setups, the time cost of A-BFD is quite low.

6.4 Experiments

Two experiments are constructed for evaluating the performance of ORAPA with different implementations. Experiment group 1 aims to compare the performance of different methods of solving SMKP under different problem configurations. Experiment group 2 focuses on assessing the performance of the integrated framework for solving the online BPP.

6.4.1 Experiment Group 1: Solving Stochastic Variable-Sized Bin Packing Problem

Experiment Group 1 aims to compare the performance of the algorithms proposed in Section 6.3.4, namely those based on Gurobi, FGA, and Monte Carlo-enhanced A-BFD. Three experiments have been devised to assess the algorithms' efficacy. In each experiment, only one variable is altered, and the performance of each

method's optimal solution on unknown instances is recorded. Specifically, given the distribution of items, two sets of instances are sampled. The first set consists of 10 instances, which will be utilised for constructing the Monte Carlo simulation as defined in Eq.(6.5)-(6.9). The second set also comprises 10 instances and will be used to evaluate the performance of the new remnant pattern obtained after solving the Stochastic Multiple Knapsack Problem. The average value of Eq.(6.5) on the test instances is calculated to gauge the algorithm's performance on unknown instances. This approach ensures the robustness of the results obtained with respect to unknown instances. In this group of experiments, the FGA is configured with a population of $N_p = 20$ and a maximum generation of $N_g = 50$. The probability of crossover is set to $p_c = 0.2$, and the probability of mutation is set to $p_m = 0.2$.

Next, the detailed configurations of the experiments are presented. In all three experiments, the item types are set from 1..99, and the maximum bin capacity is set to 100. The capacity of each bin is established through uniform sampling. In the first experiment, instances containing a total of 50 items are considered, and the total number of bins is varied. This variation will lead to changes in the tightness of the bin capacity constraints, ranging from tight to relaxed. In the second experiment, the total number of bins is fixed at 30, and the total number of items is modified. This adjustment will result in a transition of item types needing to be packed from sparse to dense. Lastly, in the third experiment, the quantity of items and bins is maintained, with different item distributions being employed, including the Normal distribution with parameters $\mu = 50$ and $\sigma = 20$, the DualNormal distribution with $\mu = (25, 75)$ and $\sigma = (20, 20)$, and the Weibull distribution with shape parameter $k = 2$ and scale parameter $\lambda = 30$.

Table 6.1 presents the results of the three sets of experiments. As the number of available bins increases, more high-value items are included in the solution of the Stochastic Multiple Knapsack Problem, reflected in the updated pattern,

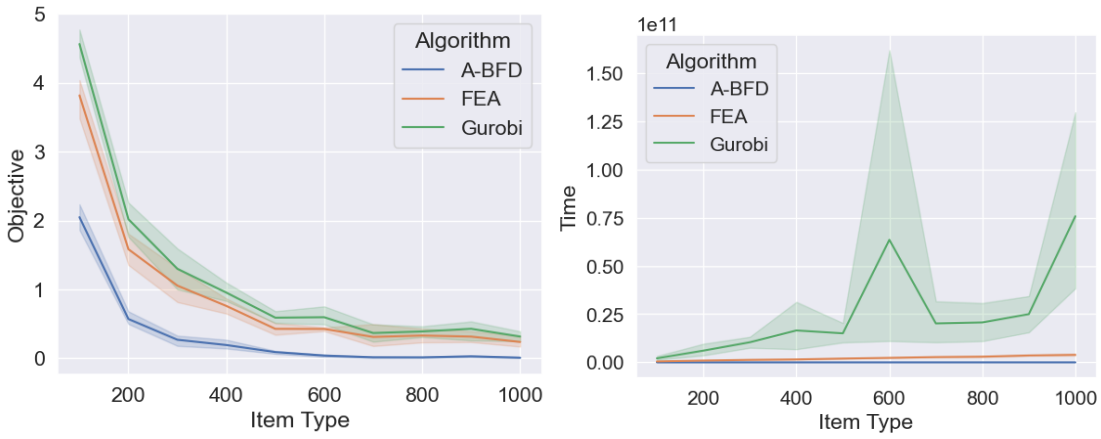
Different No. Bins with $N = 50$						
	$U = 10$		$U = 30$		$U = 50$	
	Obj	Time(s)	Obj	Time(s)	Obj	Time(s)
A-BFD	0.93	0.01	1.85	0.01	2.69	0.01
Gurobi	**2.45**	0.29	**4.09**	0.69	**6.22**	0.76
FGA	1.65	0.44	3.77	0.51	5.65	0.46
Different No. Items with $U = 30$						
	$N = 10$		$N = 30$		$N = 50$	
	Obj	Time(s)	Obj	Time(s)	Obj	Time(s)
A-BFD	0.06	0.01	0.87	0.01	1.85	0.01
Gurobi	**0.41**	0.28	**2.46**	0.70	**4.09**	0.69
FGA	0.32	0.49	2.00	0.52	3.77	0.51
Different Distributions with $N = 50$ and $U = 30$						
	Normal		DualNormal		Weibull	
	Obj	Time(s)	Obj	Time(s)	Obj	Time(s)
A-BFD	2.97	0.01	0.14	0.01	3.52	0.01
Gurobi	**6.01**	0.68	**0.78**	0.27	**6.55**	> 30
FGA	5.55	0.45	0.74	0.46	6.05	0.51

Table 6.1: Total item rejection under uniform distribution with different numbers of items and bins

thereby showing a trend of increasing objective values with the growing number of bins. Conversely, keeping the available bin quantity constant while increasing the number of items leads to a rise in the demand for high-value items, thereby elevating the objective function value. Given a fixed number of bins and items, the varying values of different items under different distributions result in diverse objective function values.

Across all experiments, the Gurobi solver achieved the best objective function values and demonstrated reasonable computation times in most cases. However, unexpectedly, under the Weibull distribution, the Gurobi solver exhibited extremely long computation times. This could be attributed to the concentration of this distribution's peak on small items, leading to a significant increase in feasible solutions and consequently greatly expanding the solver's search space. The A-BFD method showed the best execution efficiency but yielded poorer solution quality, even under the best circumstances, falling short of the Gurobi solver's performance by more than 50%. In contrast, the FGA method achieved performance with an

average difference of only 13% from the Gurobi solver's solution within an average execution time of 0.5 seconds.



(a) (b)

Figure 6.2: Results when changing the number of item types. (a) Objective. (b) Time Cost.

The impact of varying the number of item types on algorithm performance, as mentioned in Section 6.3.4, is further discussed. Here, a configuration of $N = 50$, $U = 30$, and sampling from a uniform distribution is used. The performance of the three algorithms is compared as the number of item types is increased from 100 in increments of 100 up to 1000. Figure 6.2 illustrates the changes in objective function values and time as the number of item types is increased. From the figure, it is evident that overall, better objective function values are still achieved by the Gurobi solver. However, its computation time shows exponential growth and greater fluctuations. Conversely, relatively good performance is delivered by the FGA with an approximate linear growth in time complexity. The increase in its time complexity may be attributed to the $O(n)$ complexity of the sampling algorithm and the item-level local search operator. The time cost of A-BFD does not significantly increase with the number of item types, but high-quality solutions are difficult to attain by it.

6.4.2 Experiment Group 2: Solving Online Stochastic Bin Packing Problem

Based on the experimental results mentioned earlier, the implementation of ORAPA will be carried out based on the Gurobi solver (ORAPA-GRB) and the FGA (ORAPA-FGA), which will be integrated into the complete Plan-and-Pack framework. A comparison will be made between the proposed approach and the typical zero-knowledge algorithm Best Fit, the recently proposed prediction-based algorithms ProfilePack and PatternPack, as well as the plan-and-pack algorithm Column Generation Plan-and-Pack without a tactical stage. The CGPP is configured with parameters kept the same as [Zhang et al., 2024]. For ORAPA, the planning stage and packing stage remain consistent with the implementation in (Column Generation Plan-and-Pack). Most parameters are kept unchanged, but the overestimate tolerance θ_o is increased to 2, causing the algorithm to lean more towards using patterns rather than the zero-knowledge fallback heuristic. Additionally, a maximum of 30 bins is allowed to be selected for pattern updating. The ProfilePack algorithm is configured with the parameter $\lambda_{pp} = 0.5$, as proposed by [Angelopoulos et al., 2022], and PatternPack follows the configuration delineated in [Lin et al., 2022]. The memory window's length for both is set to $k_{PrP} = k_{PaP} = 500$, the same as the parameters in the previous works [Zhang et al., 2024].

The comprehensive performance of the proposed method will first be tested under relatively complex distributions. Initially, the DualNormal distribution proposed by Burke et al. [2010] and the Weibull distribution commonly found in cloud computing, as described in [Castiñeiras et al., 2012], will be used to evaluate the algorithm's performance under static complex distributions. The DualNormal distribution described by Burke et al. [2010] is composed of different normal distributions with mean values $\mu = 25, 30, ..., 50$ and standard deviations $\sigma = 5, 10$. Since the original datasets with numbers 1-3 are established with a single normal

distribution, the DualNormal part, with numbers 4-11, is used. The diversity of the Weibull distribution is controlled by its shape parameter, and Weibull distributions with $\lambda = 0.5, 1.0, 1.5, 2.0, 3.0, 5.0$ are selected to construct the test datasets. Specifically, the Gurobi-based implementation of ORAPA in Experiment Group 1 showed a significant increase in time cost. For the Weibull distributions, the ORAPA-FGA is configured with a time limit of 0.5 seconds to avoid the algorithm being executed for a long time. The experimental setup is as follows: each trial will present the average gap of the L2 lower bound across 10 independently sampled instances. The range of item types will span from $1, 2, ..., 99$ with a total capacity set at 100. Each instance will consist of 20,000 items.

	Burke's DualNormal							
	No.4	No.5	No.6	No.7	No.8	No.9	No.10	No.11
BF	205.0	167.5	75.2	50.6	180.6	145.6	96.5	54.5
CGPP	115.8	82.4	53.5	37.7	102.9	80.3	57.9	42.7
ProP	260.0	202.0	196.9	120.5	198.4	165.6	215.7	185.2
PtnP	178.6	165.3	115.5	78.9	172.0	140.3	98.1	73.2
ORAPA-GRB	**100.2**	**49.7**	**22.6**	**18.6**	**91.9**	**49.5**	**29.8**	**21.6**
ORAPA-FGA	110.0	63.0	28.2	21.0	96.0	54.2	34.8	24.2

Table 6.2: The average objective gaps to L2 bound on Burke's DualNormal Distributions [Burke et al., 2010].

	Weibull Distribution					
	$\lambda = 0.5$	1.0	1.5	2.0	3.0	5.0
BF	**0.2**	33.0	125.3	215.4	324.1	429.1
CGPP	130.2	34.1	62.8	64.4	116.5	147.1
ProP	2459.6	475.4	341.1	345.8	303.3	500.4
PtnP	**0.2**	45.0	178.3	266.6	367.5	509.1
ORAPA-GRB	13.5	18.8	33.1	57.3	97.0	144.2
ORAPA-FGA	12.8	**13.9**	**31.7**	**56.2**	**93.6**	**143.5**

Table 6.3: The average objective gaps to L2 bound on Weibull Distributions [Castiñeiras et al., 2012].

Table 6.2 presents the numerical experimental results under the DualNormal distribution. ORAPA-GRB achieves state-of-the-art performance under this distribution; however, ORAPA-FGA also surpasses other methods. The average gap in objective function values between ORAPA-FGA and ORAPA-GRB is only around 14%, a conclusion that aligns closely with the findings from Section 6.4.1.

Table 6.3 demonstrates the experimental outcomes under the Weibull distribution. Despite both iterations of ORAPA showcasing enhanced performance, it comes as a surprise that ORAPA-FGA marginally surpasses ORAPA-GRB by an average of 6%. This hints at the FGA's ability proposed in Section 6.3.4 to exhibit a more consistent performance when computational resources are restricted. In the case of the Weibull distribution with $\lambda = 0.5$, Best Fit and PatternPack exhibited optimal results due to the prevalence of numerous small items in this distribution, aligning well with the characteristics of Best Fit heuristics. Conversely, the CGPP faced challenges in handling such scenarios, as noted in [Zhang et al., 2024]. However, even though the optimal solution was not attained, ORAPA demonstrated a robust ability to mitigate risky patterns, significantly enhancing the objective value.

Experiments are also established to assess the performance under dynamic distributions. The algorithms' configurations are inherited from the previous experiment. All experiments have the same item types and bin capacity parameters. The entire sequence of items was segmented into several equal parts, each drawn from a distinct distribution. The Normal-Sym utilises normal distributions with a constant standard deviation $\sigma = 10$, altering the mean in the sequence $25, 37.5, 50, 62.5, 75$. The Normal-PB is crafted in a similar vein, with the mean varying in the sequence $20, 30, 40, 50, 60$. Consequently, Normal-Sym is symmetrically aligned with half the capacity on a global scale, whereas Normal-Bias is not. The periodic Poisson distribution modulates the parameter μ within the range $5, 15, 25, 35, 45$. Lastly, the periodic Weibull distribution maintains $\lambda = 50$ while varying the shape parameter k in the order $0.5, 1.0, 1.5, 2, 5$.

Table 6.4 showcases the experimental results. Overall, ORAPA demonstrates relatively strong performance. Particularly, concerning the Weibull distribution, both ORAPA-GRB and ORAPA-FGA exhibit significant improvements of 32% over the previous prediction-based state-of-the-art algorithm, CGPP, for Poisson distribution, and the enhancement reaches 15%. In the case of the Normal-bias dis-

tribution, ORAPA-FGA outperforms ORAPA-GRB while showing only a slight advantage in other distributions. This may be attributed to the fact that in dynamically changing distributions, the solution's distribution needs to perform well not only within the same distribution but also remain open to potential out-of-distribution scenarios. ORAPA-GRB tends to over-optimise for the estimated distribution, leading to globally inefficient patterns, especially noticeable in the Normal-bias distribution.

	Periodic Distributions			
	Normal-Sym	Normal-Bias	Poisson	Weibull
BF	**1430.5**	1310.9	202.9	465.4
CGPP	1437.3	1302.6	177.9	296.2
ProP	2349.1	1356.8	738.0	1905.6
PtnP	1712.0	1437.7	204.9	609.2
ORAPA-GRB	1436.8	1272.7	150.3	205.3
ORAPA-FGA	1432.6	**1102.4**	**149.5**	**201.2**

Table 6.4: The average objective gaps to L2 bound by different algorithms for problems with dynamic distributions

The performances of CGPP and ORAPA are assessed under different item types and distributional prediction accuracy. Each algorithm has two kinds of variants: one with an exact distribution known beforehand and one with no exact distribution information.

In this set of experiments, the dataset is focused on uniform and normal distributions, with the settings of item types being varied while the bin capacity is maintained at 100. Initially, the classic configuration is considered, where item types are selected from the discrete set $1, 2, ..., 99$, ensuring all items have sizes smaller than the bin capacity. Subsequently, scenarios with more limited item type ranges are investigated. Biased item types are defined as the discrete set $10, 11, ..., 60$, while symmetric item types are defined as $25, 26, ..., 75$. Finally, even sparser item types are explored, corresponding to the discrete set $10, 20, 30, 40, 50, 60, 70, 80, 90$.

Table 6.5 encapsulates the experimental outcomes. The bold text represents the best result grouped by whether the perfect prediction is accessible. When the pre-

Uniform Distribution					
		Standard	Biased	Symmetric	Coarse
Perfect pred.	CGPP	**75.8**	39.9	**88.4**	**43.0**
	ORAPA-GRB	134.2	**34.6**	97.2	59.3
	ORAPA-FGA	124.8	47.4	112.2	52.5
Imperfect pred.	CGPP	188.4	57.0	215.2	**61.3**
	ORAPA-GRB	185.4	**47.8**	**137.6**	71.5
	ORAPA-FGA	**171.6**	54.1	162.2	63.2
Normal Distribution					
		Standard	Biased	Symmetric	Coarse
Perfect pred.	CGPP	**509.9**	832.2	157.6	489.7
	ORAPA-GRB	520.1	**814.4**	**78.2**	489.7
	ORAPA-FGA	520.8	821.1	96.4	489.7
Imperfect pred.	CGPP	537.1	954.7	167.2	489.7
	ORAPA-GRB	**529.8**	**841.6**	**100.4**	489.7
	ORAPA-FGA	534.1	857.2	131.8	489.7

Table 6.5: The average objective gaps to L2 bound on Uniform and Normal Distribution with different configuration

diction is perfect, CGPP achieves better performance than ORAPA. In contrast, both implementations of ORAPA showed improvement on standard, biased, and symmetric item type variances. For coarse item types, when the items are uniformly distributed, ORAPA failed to improve CGPP, while in normal distribution, all methods achieved the same solution quality.

Overall, ORAPA improved CGPP when the prediction was not perfect in most cases. Additionally, ORAPA showed more significant improvement with imperfect prediction on datasets with biased and symmetric item types, while on datasets with standard and coarse item types, the improvement is limited. Among the two implementations of ORAPA, the Gurobi solution outperforms FGA in most cases, but FGA showed better performance on uniform distribution with standard and coarse item types.

6.5 Discussion and Analysis of ORAPA

The experimental results in the above sections indicate that ORAPA performs well under complex and dynamic distributions. Under the setting of randomly initialised predictions, for static complex distributions, in most cases, ORAPA demonstrates performance advantages over the compared methods, especially over the CGPP without a tactical stage. In dynamic distributions, ORAPA also outperforms CGPP. It is also highlighted that both proposed implementations, ORAPA-GRB and ORAPA-FGA, achieve these advantages. This emphasises the effectiveness of introducing the ORAPA approach to the plan-and-pack framework. Through the additional tactical stage, high-risk remnant patterns in the online packing process can be promptly identified, and better replacement patterns can be discovered based on the latest observations and predictions.

It is also worth noting that Zhang et al. [2024] (Section 5.5.1) indicated that for the CGPP without ORAPA, the optimal range for the overestimate tolerance typically falls between 0.75 and 1.25, with an increase to 2 often resulting in decreased performance. Therefore, in the aforementioned experiments, the general improvement in the objective function value by ORAPA mainly stems from its effective risk assessment and pattern update mechanism, rather than solely being attributed to the influence of parameters.

Compared to widely adopted rolling-horizon methods for handling dynamic distributions, ORAPA is more lightweight. The rolling-horizon methods usually involve solving the same problem with different configurations multiple times [Glomb et al., 2022]. Such a method usually requires more time and may not be suitable for problems with heterogeneous sub-problems like online stochastic bin packing problems with patterns. In scenarios with time constraints, both ORAPA-FGA and ORAPA-GRB can efficiently produce solutions, thereby enhancing the overall performance of the algorithm. Furthermore, on average, the number of bins modi-

fied each time and the frequency of ORAPA execution are relatively low. This not only limits the time required to solve the online problem but also balances the performance between long-term and short-term prediction. Through minor pattern adjustments, ORAPA can achieve a balance between long-term and short-term benefits.

On the other hand, Table 6.5 illustrates some limitations of the proposed method. Firstly, in situations where there is good prior knowledge and accurate predictions, Online Risk-Aware Pattern Adjustment does not always perform well. Additionally, in problem settings where reaching the optimal solution is easy and the optimisation space is relatively small, Online Risk-Aware Pattern Adjustment may also achieve suboptimal solutions. This is primarily because prediction based on the latest observations often introduces noise and exhibits short-sighted behaviour, thereby impacting performance.

A further investigation is conducted into how the performance of the plan-and-pack framework is improved by ORAPA through the introduction of an additional tactical stage. Yao [1980] categorised items for bin packing based on their size relative to the bins. Specifically, items with sizes in the range $(1/2, 1]$ are classified as large items, items in the range $(0, 1/3]$ are considered small items, mid-large items fall within the range $(2/5, 1/2]$, and mid-small items range from $(1/3, 2/5]$. Biased and symmetric item types contain a higher proportion of mid-small/mid-large items and fewer small and large items. In practical terms, large items typically require an entire bin for themselves, while small items are prone to be allocated to bins with limited space during the operational stage. The superior performance of ORAPA in these configurations is attributed to its enhanced capability to accurately assess risks associated with these items and to execute more effective planning strategies, thereby optimising overall performance.

In most instances, Gurobi provides higher-quality solutions, benefiting from the efficiency of commercial solvers and their superior models. On the other hand,

FGA does not always perform worse than GRB. Regarding time constraints, FGA demonstrates more stable performance and can avoid the exponential growth of the search space with an increasing number of item types. In terms of objective function values, although FGA often obtains suboptimal solutions, the difference in quality from Gurobi solutions is minimal in most cases. Additionally, FGA performs better than Gurobi in complex distributions like Weibull. This can be primarily attributed to the local search operator tailored to the problem settings, ensuring algorithmic search within the feasible solution space and reducing the complexity of the search space. For problems with strict time constraints, complexity, or difficulties in modeling, metaheuristic methods are more suitable. Conversely, in cases where problems can be well modeled, Gurobi typically holds an advantage.

6.6 Chapter Summary

This chapter proposes ORAPA inspired by an analysis of the shortcomings of existing methods for handling complex dynamic distributions and online learning. Given the sensitivity of shadow prices to changes in item quantity constraints, this strategy employs before-packing adjustments to avoid the timeliness issues of rolling-horizon responses and the shortsightedness of on-packing uncertainty handling with fallback heuristics. Utilising an online pricing mechanism to accurately identify risky remnant patterns reduces computational demands while ensuring effective long-term planning. Experimental results demonstrate that the proposed ORAPA strategy significantly enhances the performance of the plan-and-pack framework under complex, dynamic distributions. However, in cases of completely known distributions, ORAPA may still introduce noise, leading to a decline in performance.

Chapter 7

Migrating to Solve Operation Room Scheduling Problem

7.1 Introduction

This chapter extends the study to Operation Room Scheduling Problem (ORSP), a scheduling problem that can be modeled as BPP. The operation room is the most valuable facility in hospitals [Macario et al., 1995, Cardoen et al., 2010], which underscores the criticality of surgery management for healthcare and medical institutions. Moreover, the global scarcity of medical resources during and after the pandemic highlights the necessity of research on surgery scheduling to balance the limited surgical resources and increasing patients' needs. The topic of operation room planning and scheduling covers a wide range of research topics, which can be divided into strategic, tactical, and operational stages. The strategic stage [Hall, 2012] usually involves long-term planning and may address financial and resource supply issues, such as the capacity of healthcare facilities. The tactical stage [Ma and Demeulemeester, 2013] provides mid-term guidelines for operations based on decisions made at the strategic stage. The operational stage [Díaz-López et al.,

2018] focuses on short-term surgery scheduling given fixed medical resources. To improve resource usage efficiency, enhance patient satisfaction, and control overall costs, patient cases and related surgeries are often aggregated based on type and patient conditions. This method, known as case mix, is widely applied for healthcare management and insurance payment [Fetter et al., 1980]. The design of the case mix belongs to the strategic stage [Zhu et al., 2019], while the related scheduling problems are more tactical or operational in nature.

Recent research has highlighted uncertainty as the most significant challenge in Operation Room Scheduling Problem (ORSP) [Harris and Claudio, 2022]. Uncertainty in ORSP arises from various sources: the duration of surgeries is not deterministic [Samudra et al., 2016, Gul et al., 2015], unexpected emergent surgeries or cancellations can occur randomly [Zonderland et al., 2010], and there is uncertainty in both resources and demand [McManus et al., 2003]. Among these factors, a major focus of the discussion on uncertainty lies in the stochastic nature of surgery duration. While some studies treat the surgery duration as deterministic to simplify the multi-objective [Wu et al., 2019] or multi-stage [Zhu et al., 2020] modelling, research on uncertain duration is still actively pursued by most researchers [Zhang et al., 2020, Rachuba and Werners, 2017, Berg and Denton, 2017].

Mathematical programming is widely applied to ORSP to analyze the mathematical properties of the problem, including uncertainty [Berg and Denton, 2017]. However, the NP-hardness of the problem [Zhu et al., 2019] leads to high computational complexity. Consequently, some models can only handle a limited problem size [Pang et al., 2019] or use heuristic approximation methods [Gul, 2018, Berg and Denton, 2017] to obtain an acceptable solution within a limited computational time. Metaheuristics such as evolutionary algorithms [Guido and Conforti, 2017, Runarsson and Sigurpalsson, 2019] and ant colony algorithms [Xiang et al., 2015] are applied to solve such problems. However, these methods require complete plan

generation, which necessitates rescheduling in emergency situations.

This chapter proposes a two-stage heuristic called CGAA, which combines the tactical plan and operational scheduling together, aiming to gain benefit from high-quality planning by solving mathematical modelling and being flexible on the operational stage. The tactical planning is modelled similarly to bin packing inspired modelling [Vancroonenburg et al., 2015, Li et al., 2016]. However, the uncertainty on surgery duration is decomposed and bounded by time intervals. The tactical planning problem is solved by the well-studied column generation method [Delorme et al., 2016]. In the operational stage, a heuristic based on a tactical stage plan is designed to balance executing the plan and handling uncertainty on unexpected handling or additional surgeries. Specifically, such an algorithm is robust to a semi-optimal tactical plan where the exact number of surgeries is not required. The two-stage algorithm is tested using data sampled from a published benchmark [Leeftink and Hans, 2018], which contains real-world data from hospitals in the Netherlands and generated theoretical cases. The algorithm generally outperforms the heuristic benchmarks provided by Leeftink and Hans [2018] regarding total overworking time and idle time.

7.2 ORSP Modelling

The operation room scheduling problem addressed in this chapter is based on case mix, meaning that although the durations of some surgeries may vary, they belong to the same case type and share the same distribution. Let's assume there are N surgeries to be scheduled, each with an uncertain surgery duration s_i.

The surgery execution time is modelled using a 3LogN distribution [Stepaniak et al., 2010], which is characterised by three parameters: α, ϵ, and β. Here, α represents the logarithmic procedure time, ϵ represents the error level, and β is a positive shift parameter.

$$\mu = \beta + e^{\alpha + \frac{\epsilon^2}{2}} \tag{7.1}$$

$$\sigma = \sqrt{(e^{\epsilon^2} - 1)e^{2\alpha + \epsilon^2}} \tag{7.2}$$

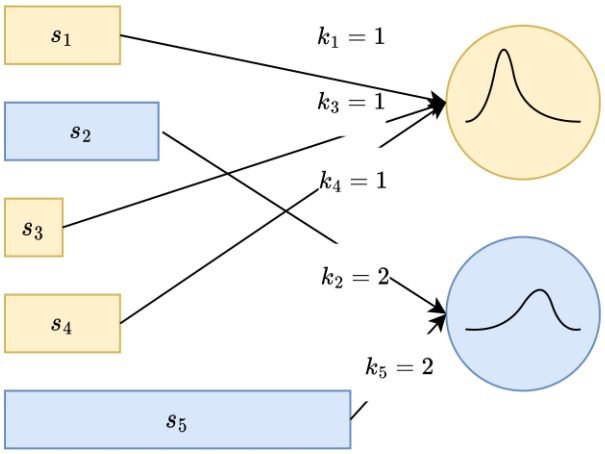


Figure 7.1: Different surgery durations might belong to the same surgery case. Surgery 1, 3, 4 belongs to case 1; 2, 5 belongs to case 2. Case 1 has a lower mean and standard deviation compared with case 2.

Let $k_i \in \{1, ..., K\}$ represent the associated case type index of surgery i as Figure 7.1 illustrates. The distribution is represented by $3LogN(k_i)$, characterised by parameters $\alpha_k, \epsilon_k, \beta_k$.

The case mixed operation room scheduling problem can be formulated as follows:

$$\min \sum_{j=1}^{M} (I_j + O_j) \tag{7.3}$$

$$I_j = \max(C - \sum_{i=1}^{N} X_{ij} s_i, 0) \qquad \forall j = 1, ..., M \tag{7.4}$$

$$O_j = \max(\sum_{i=1}^{N} X_{ij} s_i - C, 0) \qquad \forall j = 1, ..., M \tag{7.5}$$

$$\texttt{s.t.} \sum_{j=1}^{M} X_{ij} = 1 \qquad \forall i = 1, ..., N \tag{7.6}$$

$$X_{ij} \in \{0, 1\} \tag{7.7}$$

$$s_i \sim 3LogN(k_i) \tag{7.8}$$

The objective of the case-mixed operating room scheduling problem is to minimise the sum of the total idle time and overtime. The binary decision variable X_{ij} represents whether surgery i should be executed on the j-th room-day. Let I_j be the idle time and O_j be the overtime for room-day j. The idle time I_j is calculated as the capacity minus total working hours for room-day j as Equation (7.4). The overtime O_j is the duration extending beyond the capacity for room-day j as Equation 7.5. Constraint (7.6) ensures each surgery is assigned to exactly one room-day, meaning each surgery will be executed only once in the plan. Note that empty room-days are ignored in this formulation for simplicity, and the total number of non-empty room-days is denoted by M.

7.3 Related Works for ORSP

Motivated by the aim of enhancing medical management efficiency, the operations research community has attracted increasing attention over the past few

decades. Recent literature reviews highlight the diverse nature of the problem, arising from its complex features. Samudra et al. [2016] classified these problem features based on patient type, performance measures, decision level, uncertainty, and up/downstream components. Harris and Claudio [2022] conducted a related study, reviewing recent papers and expanding the taxonomy research by quantifying the complexity level of each classification aspect. The authors assert that the problem's complexity level is rising in current research trends. This increase is often achieved by considering multiple units of up/downstream, divergent interest entities, or different uncertainty resources.

Zhu et al. [2020] enhanced the discourse on problem modelling and solution methods. They highlighted the widespread adoption of a bin-packing-like formulation for multiple operating rooms, resulting in practical and satisfactory performance. The standard 1D bin packing problem aims to minimise the number of bins required for packing a series of fixed-sized items [Scheithauer, 2018], which is proven to be NP-Hard. The standard version of bin packing is considered offline, meaning the entire item sequence is known when making decisions. Conversely, the online bin packing problem is blind to the incoming item sequence and requires immediate packing decisions upon item arrival [Erdogan and Denton, 2011]. As demonstrated in Section 7.2, the formulation can also be interpreted as a bin-packing-like formulation. In this context, homogeneous room-days are treated as bins with soft capacity constraints, the surgery duration is regarded as the item, and the allocation of a surgery corresponds to the packing of an item into a bin.

As one of the most well-studied combinatorial optimisation problems, many research works have focused on both exact and approximate solutions to bin packing problems. The methodologies for solving this problem can be classified into two categories: approximation methods and exact methods. Heuristics are widely adopted since they can achieve guaranteed performance and computational complexity, making them popular in operating room scheduling. Coffman et al. [2013]

reviewed common heuristics, which are considered the most classic approximation solutions.

In the context of operating room scheduling, Vijayakumar et al. [2013] formulated the problem as a dual bin packing problem with doctor constraints. They solved it using the First-Fit-Decreasing approximation heuristic while considering different performance measures to adapt to various scenarios. This method is offline, relying on complete information about incoming surgeries and other resources, and does not consider uncertainty.

An example of addressing uncertainty in surgical scheduling is presented by Berg and Denton [2017]. They formulated the problem as a two-stage stochastic bin packing problem. Due to the NP-hardness of solving the model, they proposed a list-based heuristic approximation method that can adapt to different scenarios involving varying numbers of surgeries and surgery duration.

Metaheuristic-based methods are also valuable for tackling multi-objective problems or complex scenarios that involve multiple up/downstream units. Runarsson and Sigurpalsson [2019] employed an evolutionary algorithm to search for the optimal surgery plan. They considered fixed operation rooms over multiple weeks and incorporated duration uncertainty; therefore, the evolution algorithm is verified with the Monte-Carlo simulation. Furthermore, Nyman and Ripon [2018] compared several Metaheuristic methods for operation room scheduling using a multi-objective formulation. They explored applying different metaheuristic methods to address the scheduling problem with multiple objectives.

Although exact algorithms are usually difficult to solve due to the complexity of the operation room scheduling problem, attempts have been made to address small-scale problems or combine them with other methods. Wang et al. [2014] developed a column-generation heuristic to solve the operation room scheduling problem with cancellations. On the other hand, Sagnol et al. [2018] proposed and

solved a robust optimisation model for the problem, considering that the surgery duration follows a lognormal distribution.

It is also observed that many of the cited works mentioned above utilise private databases, often obtained through collaborations between hospitals and research institutions [Wang et al., 2014, Runarsson and Sigurpalsson, 2019, Sagnol et al., 2018]. However, a common benchmark database has been recently proposed by Leeftink and Hans [2018]. This suggests that data in the field of operation room scheduling may not have been widely available until recently [Harris and Claudio, 2022].

7.4 Column-Generation Adaptive Allocation

As discussed by Harris and Claudio [2022], dealing with uncertain surgery duration and arrival times is considered to be one of the most challenging aspects of operating room scheduling. Both the tactical and operational stages are affected by duration uncertainty. In the operational stage, potential cancellations and emergent surgeries can disrupt the plans made during the tactical stage. To address the complexity of duration uncertainty, a common approach is to estimate the duration using the mean duration of each case [Vijayakumar et al., 2013, Calegari et al., 2020], which does not handle the variance of data. When considering cancellations or emergency surgeries, rescheduling strategies are often employed [Addis et al., 2016]. However, these strategies often involve breaking the original plan and may not respond to changes in time. To tackle these challenges, a hybrid two-stage algorithm CGAA is proposed to solve the operating room scheduling problem, encompassing both tactical and operational decision-making levels.

In a broad sense, this two-phase algorithm aligns with the plan-and-pack framework proposed in earlier chapters. Considering the scenarios of additional and canceled surgeries, although surgical scheduling does not require real-time decision-

making, the two-phase approach remains applicable for handling such unexpected situations. This allows surgeries to be appropriately scheduled even in the presence of increased uncertainty, thereby balancing idle and working time.

7.4.1 Time interval

Traditionally, the uncertainty of surgery duration can be estimated by the mean duration of the case. This method is usually unable to capture the full fluctuation range of surgery duration. Figure 7.2 shows the estimated distribution of historical data from a given case mix and associated mean. The long-tailed distribution and peaks near 250 minutes and 500 minutes would be lost by simply using the case mean to estimate the surgery duration.

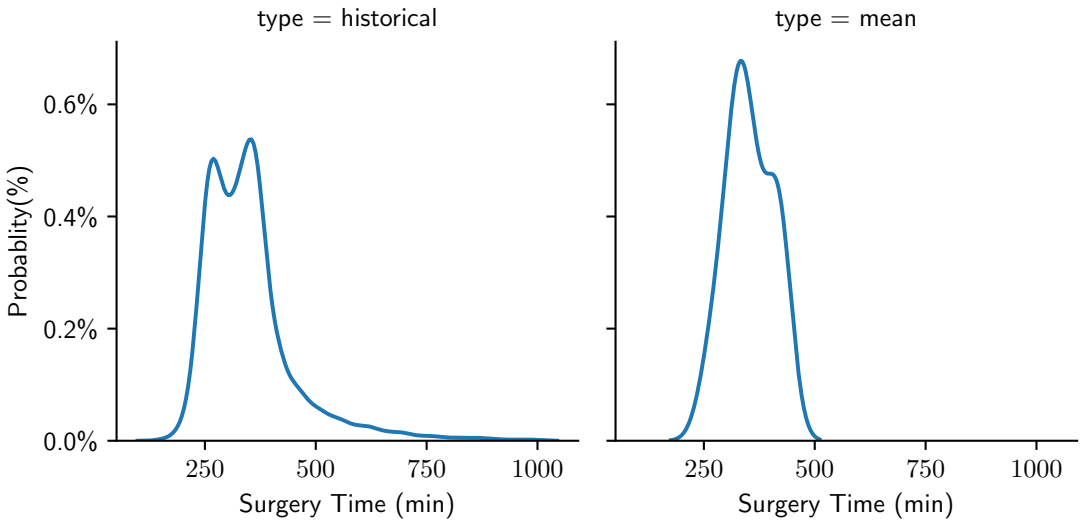


Figure 7.2: Estimated probability distribution function(PDF) curve of case TCM4 from Leeftink and Hans [2018]. Left: distribution estimated with historical data. Right: distribution estimated by all means of cases. X-axis: surgery time with minutes. Y-axis: density of PDF.

Instead of using the mean of case distribution, a time interval-based discrete distribution is proposed to represent the duration of stochastic surgery. The time domain is split into T non-overlapping time intervals $[v_t, v_{t+1})$, $t = \{1, 2, ..., T+1\}$. All time intervals share the same length for $t = 2, ..., T$, specifically $v_1 = 0$ and $v_{t+1} = \infty^+$ to capture the extremely short or long part. For each interval, the

surgery duration is estimated by:

$$e(t) = \begin{cases} v_t & \texttt{if } t = T \\ v_{t+1} & \texttt{otherwise} \end{cases} \tag{7.9}$$

Consider one surgery case $k \in 1..K$; the discrete weight function for each time interval is defined as $w_k(t)$. Therefore, the possibility mass function can be written as $p_k(t) = w_k(t) / \sum_{t'=1}^{T} w_k(t')$. Given the surgery list to be executed and the weight function for each surgery case, one can estimate the total time distribution. Denoting the number of each surgery case in the surgery list as n_k, the total surgery duration distribution to be executed is the weighted aggregation of each surgery case:

$$p(t) = \frac{\sum_{k=1}^{K} n_k w_k(t)}{\sum_{t'=1}^{T} \sum_{k=1}^{K} n_k w_k(t')} \tag{7.10}$$

7.4.2 Column generation planning

Solving the problem as directly defined by (7.3)-(7.8) is hard. Therefore, a column-generation-based approach is applied to generate a surgery plan, widely adopted in solving large-scale linear integer programming [Desrosiers and Lübbecke, 2005]. This method decomposes the original problem and iteratively improves the sub-solution with the guidance of the shadow price.

Given N surgeries and the total duration distribution, the demand for time interval t can be calculated as $q_t = \lceil N \cdot p(t) \rceil$. The tactical planning problem can be re-formulated as minimising the necessary room-days required to satisfy the surgery demands:

$$\min \sum_{p \in P} y_p \tag{7.11}$$

$$s.t. \sum_{p \in P} y_p c_p^t \geq q_t, \forall t = 1, ..., T \tag{7.12}$$

$$y_p \in \mathbb{N} \tag{7.13}$$

where $\mathcal{P} = \mathbf{p} | \sum_{t=1}^{T} c_p^t e(t) \leq C$ is the pattern set of surgery allocation of one room-day. c_p^t denotes how many surgeries of interval t should be allocated to pattern $\mathbf{p} \in \mathcal{P}$. Each pattern can be represented by a vector $\mathbf{p} = (c_p^1, c_p^2, ..., c_p^T)$. $y_{\mathbf{p}}$ is decision variable, indicating how many times pattern $\mathbf{p}$ has been used in the final plan.

The problem defined above is solved in two stages. First, a sufficiently good subset will be determined since obtaining all possible patterns is not desirable. Then, when the subset is determined, integer programming problem (7.11)-(7.13) will be solved to finalise the plan. The pattern subset will initially be set to be trivial. The restricted master problem (RMP) is obtained by relaxing the integer constraint (7.13) to be a non-negative real number. By solving RMP, one can obtain each time interval's shadow price δ_t. The shadow price represents how the constraints influence the objective value. In this problem, it is interpreted as the level of satisfactory surgery at the associated interval.

The pattern set is then improved by adding a new pattern that maximises the shadow price while satisfying the pattern set constraint:

$$\max \sum_{t=1}^{T} \delta_t c_t \tag{7.14}$$

$$\texttt{s.t.} \sum_{t=1}^{T} e(t) c_t \leq C \tag{7.15}$$

A new pattern $\mathbf{p}^*$ is obtained by solving (7.14)-(7.15), a typical knapsack problem. This problem is easy to solve when T is relatively limited. Then, the pattern subset is updated by adding the new pattern. The objective function can be transformed into the reduced cost $1 - \sum_{t=1}^{T} \delta_t c_t$. When the reduced cost is less than or equal to zero, the problem cannot be improved by adjusting the pattern subset. Therefore, the reduced cost is used to identify the end of the iteration. Algorithm 6 shows the pseudo-code of the iterative planning algorithm. The final objective value of the master problem is regarded as the expected room-days. The plan $Pl = (\mathbf{p}, y_p)|\mathbf{p} \in \mathcal{P}$ is stored as a tuple of pattern and associated adoption frequency for the operational stage.

Algorithm 6 Column Generation Tactical Planning

Input: Surgery time interval demands $\mathbf{d}$, interval duration estimation $e(t)$, room-day capacity C.
Output: Plan Pl.

1: Initialise pattern set $\mathcal{P}$
2: Set reduced cost $r \leftarrow \infty$
3: **while** $r > 0$ **do** ▷ Iteratively Generate Surgery Plan
4: Obtain all shadow price δ_t by solve RMP
5: Solve the knapsack problem (7.14)-(7.15), gain optimal pattern $\mathbf{p}^*$
6: Update pattern set $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathbf{p}^*\}$
7: $r \leftarrow 1 - \sum_{t=1}^{T} \delta_t c_t$
8: **end while**
9: Solve master problem (7.11)-(7.13), obtain Pl ▷ Gain Final Surgery Plan
return Pl

7.4.3 Adaptive operational allocation

As discussed in previous sections, the operational stage may involve two uncertain variables: the surgery duration and the random arrival or cancellation of surgeries. By applying time interval-based planning, the inconsistency of surgery demands represents the two kinds of uncertainty. For example, when the surgeries take longer than expected, the demand for a high duration will be underestimated, i.e., the high time interval demand will be less than the actual. The cancellation or emergent surgery will cause the demand estimated at the tactical stage to be underestimated or overestimated. The intuition of the operational stage heuristic is to benefit from the well-solved plan from the previous stage while being able to deal with the demand uncertainty.

Algorithm 7 represents the surgery allocation procedure. The plan Pl is generated at the tactical planning stage. Let a room-day b be a tuple $(\mathbf{s}_b, \mathbf{p}_b)$, where $\mathbf{s}_b$ represents the actual surgery assigned to room-day b and $\mathbf{p}_b$ is the associated pattern. An overestimation threshold θ is utilised to control the inconsistency between expected demand and actual demand, as in lines 5-9. This threshold allows the algorithm to keep some room-days open for possible incoming surgery while ensuring the total operating room usage rate by preventing waiting too long for overestimated surgeries.

A matching mechanism is used to determine whether surgery should be allocated to a specific room-day or not. An important assumption needs to be emphasised: the actual surgery distribution is unknown until it is completed. However, the surgery case k is known for any surgery i, whether considered during the tactical stage or randomly arriving. The time interval with the highest weight $t^* = \arg\max w_k(t)$ is used to match the pattern. Line 11 applies the matching mechanism with the pattern. When the associated count in the pattern $c^{\mathbf{P}}_{t^*} > 0$, the pattern is claimed to match the surgery. Line 9 applies the matching mechanism for the room-day,

which requires both the pattern to match the surgery and the allocated surgeries count, plus the current one, does not exceed $c_{t^*}^{\mathbf{p}}$.

As Lines 6 and 16 show, when the threshold is exceeded or the surgery is not contained in the current plan, it should be allocated using a fallback strategy. Here, a greedy heuristic is applied. For a specific room-day b, the already-allocated duration is estimated by $\sum_{t=1}^{T} c_t^b e(t)$, where c_t^b is the count of time interval t already allocated. The room-day that minimises $|C - e(t) - \sum_{t'=1}^{T} c_b^t e(t')|$ is chosen, where t is the estimated time interval given surgery case. Note that this approach might break the plan, which means a surgery might be allocated to a room-day, but the pattern associated does not match the surgery. Such breaking of rules is allowed in the proposed algorithm because it can be viewed as the result of adjusting the plan online.

Algorithm 7 Adaptive surgery allocation

Input: Surgery plan Pl, actual surgery sequence $\mathbf{s}$ with length N, overestimate tolerance threshold θ.
Output: Allocated room-days $\mathbf{b}$.

1: Initialise empty room-day list $\mathbf{b}$
2: **for** $i = 1, ..., N$ **do**
3: $t^* = \arg\max_t w_{k_i}(t)$
4: Calculate estimated remain surgery duration $s' \leftarrow \sum_{j=i}^{N} \mu(k_j)$
5: Total idle time $I = \sum_{b \in \mathbf{b}} I_b$, I_b calculated by (7.4)
6: **if** $I/s' > \theta$ **then** ▷ Too much operation room-day idle time
7: `fallback_allocation`(i)
8: Continue
9: **end if**
10: **if** $I_b \geq e(t^*) \wedge$ `match`$(b, i), \exists b \in \mathbf{b}$ **then**
11: Allocate surgery i into room-day b
12: **else if** $y_{\mathbf{p}} > 0 \wedge$ `match`$(\mathbf{p}, i), \exists \mathbf{p} \in P$ **then**
13: Add a new room-day $(\mathbf{s}_b = \vec{0}, \mathbf{p}_b = \mathbf{p})$ to $\mathbf{b}$
14: Allocate surgery i into the newly created room-day b
15: $y_{\mathbf{p}} \leftarrow y_{\mathbf{p}} - 1$
16: **else**
17: `fallback_allocation`(i)
18: **end if**
19: **end for**
return b

7.5 Experiment of Solving ORSP

7.5.1 The Leeftink Database

A numerical experiment based on the public database by Leeftink and Hans [2018] was set up. It contains 11 surgical specialties from real hospital data in the Netherlands. Apart from real-world data, they added several theoretical case-mix data based on a mixture algorithm to represent different surgical scenarios. They also provided benchmark solutions based on First-Fit-Decreasing and Best-Fit-Decreasing heuristics.

The database is categorised into different case mixes, representing real-world surgery specialties or theoretical re-mixtures. Each case-mix instance is configured using base room-days (m) and the desired workload level (l). The room-day capacity is fixed at 480 minutes. A series of surgery cases ($k \in 1, ..., K$) is then assigned to the case-mix instance, providing the distribution parameters such that $\sum_{k=1}^{K} \mu(k)/mC$ is close to l. Note that the surgeries have not been realised or executed at this point.

7.5.2 Experiment setup

Two groups of experiments were set up to provide a comprehensive view of different sources of uncertainty. Group 1 assumes that no cancellations or emergency surgeries need to be executed. Group 1 uses the surgery-case instance sampler provided by Leeftink and Hans [2018] to realise surgeries. Group 2 extends the sampling procedure by using a two-stage approach to reflect random arrivals and cancellations. A sufficient number of realisations is sampled for each surgery case using the provided sampler. First, a surgery case is uniformly selected, and then a realisation is sampled from the generated sample. This procedure maintains the

workload close to the designed level while introducing random arrival or cancellation uncertainty. Figure 7.3 represents the average difference between the desired and actual workloads on a logarithmic scale. The difference between the additional sampling and the original data is relatively small.

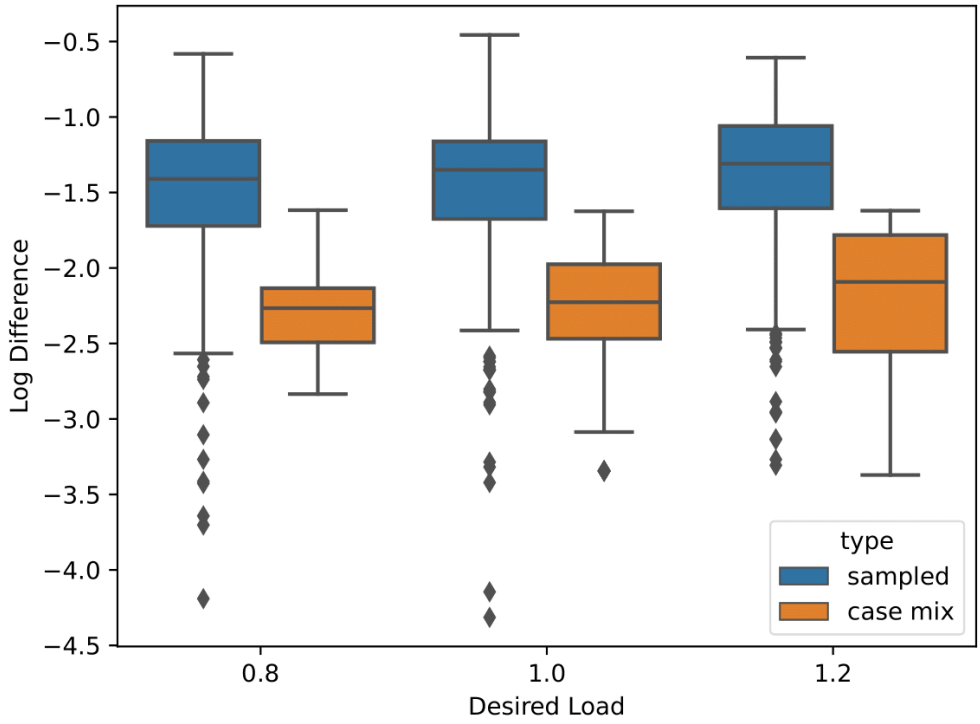


Figure 7.3: Logarithmic scaled difference between desired load and actual load. Blue: proposed sampling method. Orange: provided case mix actual load [Leeftink and Hans, 2018].

Both groups contain real-world data and theoretically generated data. As one of the largest specialties, general surgery involves a wide range of different surgery types. The real-world general surgery data (ID CHI) contain 720 case mix instances with a total of 1018 surgery types across all instances. For theoretical testing, the ID TCM4 case mix from Leeftink and Hans [2018] was used, which also includes 720 instances and 2504 generated surgery types. The generated types were selected using α and ϵ from general surgery and other specialties, including ophthalmic surgery, neurological surgery, surgical oncology, etc. The β parameter was randomly generated for simplicity.

The proposed algorithm was compared with the Best-Fit Descending (BFD) benchmark provided by Leeftink and Hans [2018], which only considers the mean of each surgery case distribution. Since we modified the surgery realisation method, the BFD benchmark has now become a tactical plan. Considering the random arrival or cancellation, surgeries that were not planned are allocated at the operational stage using a simple best-fit heuristic. For each surgery case-mix instance, 10 realisations were sampled, and the average objective value calculated using Formula (7.3) was determined. The average idle time and overtime were also calculated to investigate in detail. Due to the large database size, the results of instances were grouped with the same base room-days and desired load together to provide an overview of how the algorithm performs under specific conditions.

Most algorithm parameters were kept the same across all tested instances. The room-day capacity (C) was kept the same as defined in the database. The time intervals were set to be $[0, 60)$, $[60, 120)$, ..., $[480, \infty)$, with the same interval length for all intervals except the last one. The overestimated tolerance threshold (θ) in Algorithm 7 was set to 1 in this research. While it can be generated analytically or based on expert experience, the weight function described in Section 7.4.1 was defined using a data-driven procedure. For each surgery type, 100 additional surgery realisations were generated, and the number of surgeries falling into each interval t was counted to determine the weight: $w(t) = \texttt{count}(s_i \in [v_t, v_{t+1}))$.

7.5.3 Experiment Results

For each group and data type, the experiments were clustered based on room-days and desired load, as both factors contributed to an increase in the objective value. Three typical room-day values were selected: 5, 20, and 40, representing different short-term, middle-term, and long-term planning horizons. For each room-day type, three levels of workload were chosen to represent low (0.80), middle (1.00), and high (1.20) workloads. Three types of experimental data were

compared, including the objective value, idle time, and overtime. The column labelled "BFD" represented the results obtained using the Best-Fit-Decreasing method, while "CGAA" represented the proposed Column Generation Adaptive Allocation method described in Section 7.4. All numerical results in the tables were represented in minutes. The experimental results of the two groups were presented in Table 7.1 and Table 7.2. To facilitate discussions, each cluster was assigned a unique ID number.

Experiment Group 1: Operation Uncertainty

As shown in Table 7.1, CGAA outperformed the real-world general surgery data benchmark. In the theoretical data, CGAA outperformed most cases except when the workload was heavy in short-term and mid-term planning cases. BFD generated a better objective value in clusters TCM4-2, 3, 5, and 6 with a relatively small advantage. However, CGAA achieved significantly better objectives in both data types for long-term planning, demonstrating its potential for long-term planning and operations. While there were some extreme cases like TCM4-6 and TCM4-9, BFD tended to reduce idle time. However, this over-concentration of idle time resulted in high overtime work. CGAA also aimed to reduce idle time but exhibited more balanced behaviour and had fewer extreme cases.

Experiment Group 1: Multi-Source Uncertainty

Table 7.2 presents the results under random cancellations and additional surgeries. Surprisingly, when assuming a non-perfect surgery list, planning and operation with BFD leads to a rapid increase in both idle time and overtime, particularly in the case of real-world general surgery data. These results worsen as the planning horizon increases. In some extreme cases, such as CHI-16 18, planning and operating with BFD can result in an objective value more than double that of CGAA.

ID	RD	Load	Objective		Idle Time		Overtime	
			BFD	CGAA	BFD	CGAA	BFD	CGAA
CHI-1	5	0.80	1227.67	735.02	39.65	107.72	1188.03	627.30
CHI-2	5	1.00	1452.44	791.23	78.30	119.70	1374.14	671.53
CHI-3	5	1.20	1488.60	983.52	119.70	92.76	1368.90	890.76
CHI-4	20	0.80	2254.11	1766.24	430.21	462.27	1823.90	1303.97
CHI-5	20	1.00	2642.88	2047.28	540.23	475.23	2102.65	1572.05
CHI-6	20	1.20	2937.21	2785.46	600.25	428.38	2336.96	2357.09
CHI-7	40	0.80	3966.28	3430.60	943.32	1033.08	3022.96	2397.52
CHI-8	40	1.00	4731.33	4081.88	1218.62	1119.50	3512.71	2962.38
CHI-9	40	1.20	5452.36	5126.63	1548.29	946.23	3904.07	4180.40
TCM4-1	5	0.80	664.27	562.30	81.07	157.28	583.21	405.02
TCM4-2	5	1.00	617.18	648.66	214.77	175.31	402.41	473.35
TCM4-3	5	1.20	711.14	827.95	461.86	234.66	249.28	593.29
TCM4-4	20	0.80	2799.95	2124.30	476.11	884.68	2323.84	1239.62
TCM4-5	20	1.00	2481.01	2545.85	1187.22	1030.03	1293.79	1515.81
TCM4-6	20	1.20	3062.83	3076.73	2361.76	1190.31	701.07	1886.42
TCM4-7	40	0.80	5470.33	3935.36	730.54	1563.86	4739.79	2371.51
TCM4-8	40	1.00	4983.01	4799.59	2371.01	2120.90	2612.00	2678.69
TCM4-9	40	1.20	6049.96	5749.37	4707.84	2553.54	1342.12	3195.83

Table 7.1: Group 1: No Cancellation & Extra Surgery

This indicates that planning and operating with BFD is sensitive to cancellations and additions. On the other hand, CGAA achieves stable performance on both real-world and theoretical data.

7.6 Discussion on Performance of CGAA

The proposed CGAA algorithm generally exhibits superior performance compared to the BFD benchmark in terms of objective value and stability. When cancellations and additions are not considered, the main source of risk lies in surgery duration. As the actual surgery duration is unknown during the operational stage, it closely follows the planned duration. Therefore, Group 1 results are primarily influenced by planning process quality. The success of Group 1 results confirms that column generation planning can generate high-quality plans. It is worth noting that the time interval formulation serves as the foundation for achieving these

ID	RD	Load	Objective		Idle Time		Overtime	
			BFD	CGAA	BFD	CGAA	BFD	CGAA
CHI-10	5	0.80	2692.09	655.81	46.52	120.38	2645.57	535.43
CHI-11	5	1.00	2876.99	941.97	130.14	295.42	2746.85	646.54
CHI-12	5	1.20	3372.19	1174.74	172.85	228.52	3199.34	946.21
CHI-13	20	0.80	7479.74	2056.98	418.93	683.56	7060.80	1373.42
CHI-14	20	1.00	9047.72	2645.76	465.84	972.85	8581.89	1672.91
CHI-15	20	1.20	10719.90	3886.11	642.38	943.09	10077.52	2943.02
CHI-16	40	0.80	14070.67	3733.55	664.50	1164.74	13406.17	2568.81
CHI-17	40	1.00	17806.05	4772.65	1008.16	1756.26	16797.89	3016.39
CHI-18	40	1.20	20473.44	7683.77	1199.65	2033.62	19273.78	5650.15
TCM4-10	5	0.80	788.58	580.60	48.66	215.80	739.92	364.79
TCM4-11	5	1.00	802.68	806.91	165.89	307.48	636.79	499.43
TCM4-12	5	1.20	1046.28	961.02	279.13	407.70	767.15	553.32
TCM4-13	20	0.80	3097.64	2514.16	379.04	1137.03	2718.60	1377.13
TCM4-14	20	1.00	3422.49	3026.84	821.74	1498.11	2600.75	1528.73
TCM4-15	20	1.20	3970.49	3624.35	1217.79	1722.41	2752.70	1901.95
TCM4-16	40	0.80	5995.03	4704.83	593.90	2232.45	5401.13	2472.38
TCM4-17	40	1.00	6685.01	5901.75	1492.25	2969.88	5192.76	2931.87
TCM4-18	40	1.20	7935.96	7000.73	2786.90	3780.01	5149.06	3220.72

Table 7.2: Group 2: Random Cancellation & Extra Surgery

high-quality plans.

By incorporating uncertainty through time intervals and weight functions, CGAA enables uncertainty to be considered at the tactical stage. This allows generation of surgical plans that naturally account for uncertainty, as discussed in Section 7.4.1. Moreover, solving a vanilla stochastic integer programming model of Equations (7.3)-(7.8) is challenging, as it requires determining $M \times N$ decision variables. The time interval formulation helps decompose the problem into solvable subproblems, facilitating the optimisation process.

In Group 2 experiments, introducing uncertainty through random cancellations and additional surgeries adds complexity to operating room scheduling. The occurrence of emergency surgeries can disrupt planned schedules by consuming allocated time slots for incoming surgeries. When cancellations occur, allocated time blocks need to be reallocated, which can result in improper scheduling or unused time slots, leading to increased idle time.

As surgeries are unlikely to be shorter than 1 hour, it becomes necessary to eliminate overestimated time blocks to optimise scheduling. Group 2 results demonstrate that Algorithm 7 successfully fulfils its design purpose by effectively handling uncertainties introduced by random cancellations and additional surgeries.

Furthermore, the two-stage CGAA algorithm achieves high stability regarding data duration. Figure 7.4 illustrates the case mix contour of the two case mix types used in experiments. The x-axis represents the ratio of average surgery duration to block length (μ/C), indicating how much of the block is typically occupied by an average surgery. The y-axis represents the coefficient of variation (σ/μ), reflecting surgery duration variability.

The TCM4 case mix exhibits longer durations and lower variability, suggesting that surgeries in this case mix have relatively fixed durations. When deviations from the original plan occur, they are more likely to be resolved by substituting one surgery with another similar surgery.

Real-world general surgeries display greater duration diversity. Despite using the same sampling process for both case mixes, the high variance in real-world surgery durations is the primary factor contributing to the BFD benchmark's sensitivity and poor performance in Group 2 experiments. This demonstrates that CGAA can overcome challenges from multiple sources, including high variance in real-world surgeries.

7.7 Chapter Summary

The major challenge in operating room scheduling is rooted in the management of uncertain surgery durations and multiple sources of uncertainty inherent to the decision-making process. In this chapter, emphasis was placed on addressing the stochastic nature of surgery durations and the occurrence of random cancellations

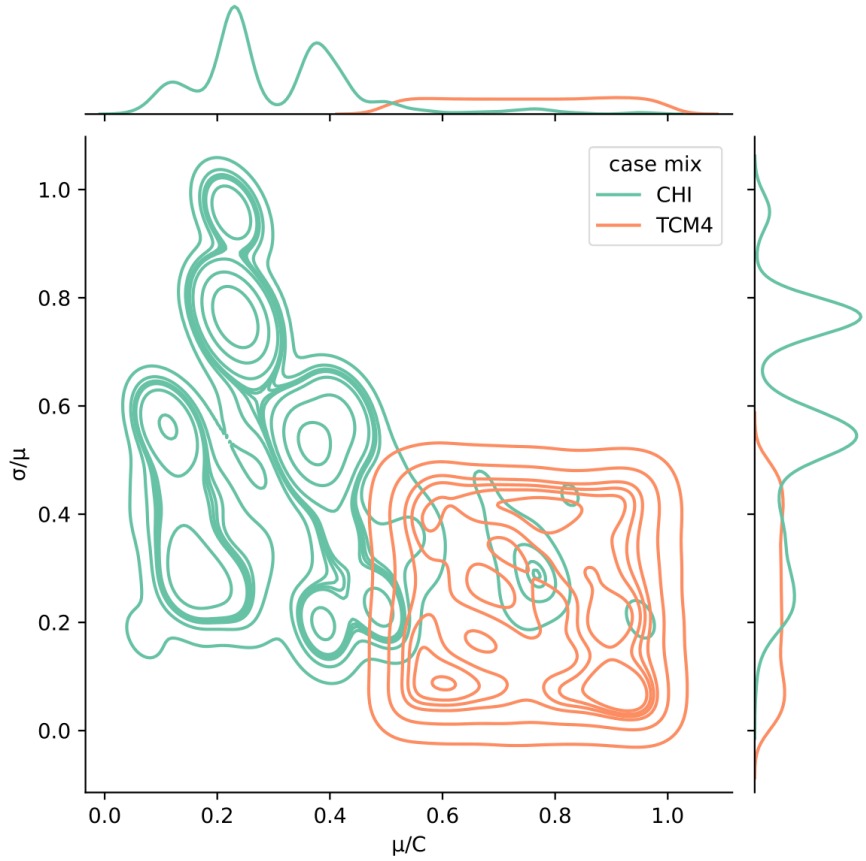


Figure 7.4: Case mix contour of case mix TCM4(Orange) and CHI(Green).

or emergent surgeries. The CGAA algorithm, a two-stage planning and operation model, was proposed. This framework was a revised version of the previously mentioned plan-and-pack approach, utilising discretisation and weight functions to address the ORSP.

The algorithm was tested with real-world surgery data and a theoretical case mix provided by Leeftink and Hans [2018]. High-quality solutions were demonstrated to be generated by CGAA, outperforming the Best-Fit-Decreasing benchmark. Furthermore, stability was exhibited by the algorithm, and solution quality was ensured across diverse problem profiles.

However, it is important to note that one limitation of our study was the limited discussion on constraints. As highlighted by Harris and Claudio [2022], there was a growing trend in incorporating more upstream and downstream constraints in

the modelling process, which requires close collaboration with hospital managers. A potential avenue for future research is to explore the integration of additional constraints while maintaining solution quality and stability under different uncertainty scenarios.

Chapter 8

Conclusions

8.1 Review of Research Contents

This thesis aims to address online packing problems through the application of pattern-based methodologies. Inspired by the widely adopted concept of patterns in machine learning and related fields, the study proposes guiding online decision-making by reusing high-quality patterns.

Comparative analysis reveals that the direct application of patterns for solution construction remains relatively underexplored in contemporary operations research. Notably, while zero-knowledge and full-knowledge approaches have been extensively investigated for packing problems, algorithms utilizing historical information for predictive half-knowledge scenarios show significant research gaps. This study conducts pioneering explorations in both dimensions.

Given the complexity and dynamic nature of real-world distributions, rigorous testing was conducted using both intricate static distributions and temporally evolving patterns. To extend the methodology's applicability, the investigation was expanded to the Operation Room Scheduling Problem, a practical scheduling problem characterised by heterogeneous uncertainty sources. The successful ap-

plication of pattern-based approaches in these diverse contexts demonstrates their generalisability and adaptability to real-world operational challenges.

The proposed methodologies in this thesis are backed by the plan-and-pack framework. This framework generates a packing plan consisting of patterns and their corresponding usage quotas by predicting item quantities over a certain period and solving accordingly. This generated packing plan is used to guide decisions in online packing. The main challenges of this framework include: 1. finding effective algorithms to generate efficient patterns, and 2. the need for online decision-making to dynamically balance plan adherence to mitigate prediction error impacts. To address these two challenges, the CGPP algorithm is designed, which ensures pattern quality through a column generation-based planning mechanism and employs on-packing heuristics to manage demand prediction errors, as well as a rolling-horizon-like replanning mechanism to adapt to dynamically changing patterns.

Despite demonstrating relatively good performance across several complex instances, the on-packing uncertainty-handling heuristics employed by CGPP still exhibit response latency weaknesses. To enhance the algorithm's ability to cope with complex and dynamic distributions in an online learning context, ORAPA has been proposed. This method improves overall algorithm performance by proactively analysing the risks associated with patterns, allowing for the early replacement of high-risk patterns.

Finally, we examine whether the proposed algorithms can address challenges encountered in practical applications. The Operation Room Scheduling Problem is chosen as the test problem. As a derivative of the BPP, the ORSP presents additional uncertainties. The proposed approach is tested using both real hospital data and synthetic data. The results indicate that the pattern-based optimisation method performs well in addressing this real-world problem.

8.2 Discussion and Findings

The proposed plan-and-pack framework differs from the classic rolling-horizon method by emphasises the importance of dynamic adjustments during execution. While both approaches can address dynamic problems, the rolling-horizon method requires completing one cycle before making adjustments. Consequently, if planning is imperfect, the rolling-horizon method often lacks the ability to respond in time. In contrast, the plan-and-pack approach eliminates potential performance declines caused by imperfect planning through online dynamic adjustments. This robustness is achieved through implementing adaptive strategies during execution.

In the realm of pattern recognition, experiments conducted in this study demonstrate the overall effectiveness of the proposed pricing-based pattern generation method. This approach not only uses shadow pricing to evaluate pattern quality but also involves progressively constructing pattern sets.

Total bin usage is a critical measure in online packing problems. Pricing-based pattern generation has shown commendable performance, particularly when provided with an accurate item distribution. In an online learning context, CGPP, as a typical prediction-based method, surpasses existing prediction-based baselines. Furthermore, this method achieves a pattern distribution closely approximating the offline optimal solution, encompassing both pattern types and frequencies. This may be attributed to the pricing-based approach's ability to account for global quantity constraints and represent them as a function of pattern quality.

From a mathematical perspective, individual shadow pricing indicates the marginal cost of changes in the number of item types corresponding to a given pattern set, specifically in terms of additional bins required. Shadow prices are sensitive to key item types influencing the objective function value. Therefore, during the iterative process of refining pattern sets, shadow price can more accurately identify high-quality patterns.

In cases of erroneous predictions regarding item types, the generated patterns may be suboptimal. For complex dynamic distributions, online learning estimates often contain errors. These errors can affect the expected quantity of item types. In pattern-based problems, overestimation is particularly likely to impact the objective function value. Patterns serve as blueprints for packing specific bins in the future. Overestimation may leave bins unfilled according to allocated patterns, leading to wasted space. Two algorithms proposed here are specifically optimised to address this. CGPP employs an on-packing strategy, which controls whether the algorithm switches to zero-knowledge heuristics based on expected incoming items. This method enforces placing some items into bins that may experience overestimation, thereby making necessary adjustments.

Due to the assumption in online bin packing that the exact future item sequence is unknown, the algorithm cannot quickly determine whether adjustments should be made when the item distribution changes. This causes potential response latency in CGPP. In contrast, ORAPA leverages shadow pricing sensitivity to item quantity constraints to assess whether a pattern may pose risks. This enables proactive analysis and replacement of high-risk patterns. Unlike CGPP's on-packing strategy, ORAPA adopts a before-packing strategy, intervening by solving a smaller-scale problem to correct high-risk patterns. This approach demonstrates superior experimental performance despite additional computational costs.

Based on conventional OBPP success, the plan-and-pack framework and pricing-based pattern generation methods apply to ORSP, demonstrating this approach's superior performance through experiments. This method also enables applying the plan-and-pack framework to similar packing problems involving uncertainty. This paradigm's core transforms uncertain items into discrete deterministic items through discretization and weighting functions. This transformation reduces multi-source uncertainty into global item quantity uncertainty, solvable using pattern-based methods, making it compatible with the proposed plan-and-pack

framework. For problem settings involving continuous items, the discretization strategy can restructure them to facilitate applying the plan-and-pack framework.

8.3 Research Limitations

This research primarily addresses two critical challenges: identifying effective patterns within an exponentially growing pattern space, and mitigating error guidance caused by prediction bias in uncertain environments. For the former challenge, a shadow price-based approach is employed to filter efficient patterns from the vast pattern space, subsequently reusing them to generate packing plans. This methodology is grounded in the sensitivity of shadow prices and their corresponding column generation method to variations in critical item types, thereby enabling more effective pattern selection while avoiding exhaustive pattern space exploration. Experimental results demonstrate the column generation method yields high-quality solutions. However, this also implies its susceptibility to prediction errors, where erroneous plans may misguide online decision-making. For the latter challenge, two distinct approaches are proposed to handle uncertainty: a threshold-based during-packing method; and a before-packing method, ORAPA, which leverages online pricing to solve the Stochastic Multiple Knapsack Problem. Although experimental results validate the efficacy of both approaches, the former relies on manual parameter tuning, potentially requiring extensive trial-and-error to identify optimal parameters while partially neglecting distribution information. The latter, while capable of utilising distribution information, incurs higher computational costs. Additionally, ORAPA exhibits vulnerability to noise interference and may underperform compared to CGPP when prior knowledge is available.

For certain specific distributions, such as the Normal Distribution with symmetric item types, both CGPP and its developed variant, ORAPA, perform poorly compared to traditional BF heuristic algorithms. On one hand, such distributions are

relatively favourable to BF: for any item type t, one can always find an equally probable item type t' with size $s_{t'} = B - s_t$, allowing these two items to be perfectly packed into one bin. On the other hand, to address potential uncertainties, the plan-and-pack approach necessitates adjustments to mitigate the impact of prediction errors, which can lead to conservative decision-making, thereby resulting in suboptimal performance on relatively straightforward distributions. This phenomenon exemplifies the well-known No-Free-Lunch Theorem.

CGPP introduces multiple algorithm parameters to tackle dynamic issues and alleviate prediction error effects. In Section 5.5.3, a quantitative analysis examines how these parameters influence objective function values. The algorithm's initial settings demonstrated superior experimental performance. However, determining appropriate parameter settings for specific tasks remains challenging, often requiring a trial-and-error approach.

Finally, although experiments indicate the proposed methods yield satisfactory results, this study lacks a complete analytical framework, particularly for asymptotic analysis. The challenge in this area arises from the organic integration of integer programming with online heuristics. The analysis of the algorithm's time complexity and asymptotic behaviour should consider both aspects simultaneously. While Chapter 5 offers preliminary insights into this issue, further research is needed to analyse the framework with the pricing-based pattern generation in greater depth.

8.4 Future Works

In this study, both the plan-and-pack framework and the plan-and-adjust-and-pack framework are implemented in a sequential manner. Since planning typically requires significantly more computational time than online decision-making, this leads to an imbalance in decision time allocation. For more complex COP in-

stances, this sequential implementation may result in excessively long decision-making processes, thereby compromising real-time performance. Parallelising planning and online decision-making processes could potentially resolve this imbalance. This approach would entail addressing several research questions, including information synchronisation and communication between planning, packing and potential adjustment modules, as well as making online decisions under conditions of less reliable planning outcomes.

With recent advancements in machine learning, particularly deep learning, integrating ML-based methods with traditional integer programming and metaheuristic algorithms to solve combinatorial optimisation problems has become a significant focus. Deep learning approaches using neural networks can achieve rapid inference speeds on modern computational hardware and have been extensively applied to pattern mining tasks in data science. In contrast, the pattern generation and planning methodology proposed here relies on solving integer programming models, which is computationally intensive. Therefore, leveraging machine learning to enhance pattern generation and potentially improve planning efficiency presents a promising research direction. Current methods still rely on solving certain offline problems. It is worth investigating whether machine learning and reinforcement learning approaches could allow for the avoidance of solving offline problems entirely or achieving complete solutions, thereby internalising planning within the decision-making process.

This thesis focuses on solving Online Bin Packing Problem and its approximate variants. Other online stochastic combinatorial optimisation and operations research problems remain unexplored. This includes problems with similar structures but different constraints and objective functions, such as job shop scheduling problems, as well as structurally distinct problems like vehicle routing problems. Whether the successful plan-and-decision framework and pricing-based pattern identification methods from OBPP can adapt to these problems remains an open

question.

It would also be beneficial to extend the pattern-based and prediction-driven plan-and-decision framework to other applications in online combinatorial optimisation problems. Potential research directions cover both planning and decision-making aspects. At the planning level, it is crucial to determine which problems need solving and whether they can be solved effectively and quickly. If not, can a proxy strategy be identified for planning? At the decision level, it is essential to explore assessing the reliability of predictions in the presence of errors and making decisions that benefit from planning while mitigating the consequences of prediction errors. Possible areas for extension include structurally similar problems such as job shop scheduling problems, higher-dimensional bin packing problems, and structurally different yet similarly challenging problems like periodic vehicle routing problems with dynamic customer demand.

Bibliography

[1] B. Addis, G. Carello, A. Grosso, and E. Tànfani. Operating room scheduling and rescheduling: A rolling horizon approach. *Flexible Services and Manufacturing Journal*, 28(1):206–232, June 2016. ISSN 1936-6590. doi: 10.1007/s10696-015-9213-7.

[2] S. Ali, A. G. Ramos, M. A. Carravilla, and J. F. Oliveira. On-line three-dimensional packing problems: A review of off-line and on-line solution approaches. *Computers & Industrial Engineering*, 168:108122, June 2022. ISSN 0360-8352. doi: 10.1016/j.cie.2022.108122.

[3] A. M. Alvarez, Q. Louveaux, and L. Wehenkel. A supervised machine learning approach to variable branching in branch-and-bound. In *In Ecml*, 2014.

[4] A. M. Alvarez, Q. Louveaux, and L. Wehenkel. A Machine Learning-Based Approximation of Strong Branching. *INFORMS Journal on Computing*, 29 (1):185–195, Jan. 2017. ISSN 1091-9856, 1526-5528. doi: 10.1287/ijoc.2016.0723.

[5] L. Andrew, S. Barman, K. Ligett, M. Lin, A. Meyerson, A. Roytman, and A. Wierman. A tale of two metrics: Simultaneous bounds on competitiveness and regret. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, pages 329–330, Pittsburgh PA USA, June 2013. ACM. ISBN 978-1-4503-1900-3. doi: 10.1145/2465529.2465533.

[6] D. Angelis, F. Sofos, and T. E. Karakasidis. Artificial Intelligence in Physical Sciences: Symbolic Regression Trends and Perspectives. *Archives of Computational Methods in Engineering*, 30(6):3845–3865, July 2023. ISSN 1886-1784. doi: 10.1007/s11831-023-09922-z.

[7] S. Angelopoulos, C. Dürr, S. Kamali, M. P. Renault, and A. Rosén. Online Bin Packing with Advice of Small Size. *Theory of Computing Systems*, 62 (8):2006–2034, Nov. 2018. ISSN 1433-0490. doi: 10.1007/s00224-018-9862-5.

[8] S. Angelopoulos, S. Kamali, and K. Shadkami. Online Bin Packing with Predictions. In *Thirty-First International Joint Conference on Artificial Intelligence*, volume 5, pages 4574–4580, July 2022. doi: 10.24963/ijcai.2022/635.

[9] S. Angelopoulos, S. Kamali, and K. Shadkami. Online Bin Packing with Predictions. *Journal of Artificial Intelligence Research*, 78:1111–1141, Dec. 2023. ISSN 1076-9757. doi: 10.1613/jair.1.14820.

[10] S. Angelopoulos, C. Dürr, S. Jin, S. Kamali, and M. Renault. Online Computation with Untrusted Advice, Apr. 2024.

[11] S. Angelopoulos, C. Dürr, S. Jin, S. Kamali, and M. Renault. Online computation with untrusted advice. *Journal of Computer and System Sciences*, 144:103545, Sept. 2024. ISSN 0022-0000. doi: 10.1016/j.jcss.2024.103545.

[12] A. Antoniadis, C. Coester, M. Elias, A. Polak, and B. Simon. Online metric algorithms with untrusted predictions. In *Proceedings of the 37th International Conference on Machine Learning*, pages 345–355. PMLR, Nov. 2020.

[13] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, Sept. 1998. ISSN 0004-5411. doi: 10.1145/290179.290180.

[14] S. Asta, E. Özcan, and A. J. Parkes. CHAMP: Creating heuristics via many parameters for online bin packing. *Expert Systems with Applications*, 63: 208–221, Nov. 2016. ISSN 0957-4174. doi: 10.1016/j.eswa.2016.07.005.

[15] G. Astolfi, F. P. C. Rezende, J. V. D. A. Porto, E. T. Matsubara, and H. Pistori. Syntactic Pattern Recognition in Computer Vision: A Systematic Review. *ACM Comput. Surv.*, 54(3):65:1–65:35, Apr. 2021. ISSN 0360-0300. doi: 10.1145/3447241.

[16] Y. Azar, I. R. Cohen, S. Kamara, and B. Shepherd. Tight bounds for online vector bin packing. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 961–970, New York, NY, USA, June 2013. Association for Computing Machinery. ISBN 978-1-4503-2029-0. doi: 10.1145/2488608.2488730.

[17] Y. Azar, I. R. Cohen, A. Fiat, and A. Roytman. Packing Small Vectors. In *Proceedings of the 2016 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 1511–1525. Society for Industrial and Applied Mathematics, Dec. 2015. doi: 10.1137/1.9781611974331.ch103.

[18] B. S. Baker, E. G. Coffman, Jr., and R. L. Rivest. Orthogonal Packings in Two Dimensions. *SIAM Journal on Computing*, 9(4):846–855, Nov. 1980. ISSN 0097-5397, 1095-7111. doi: 10.1137/0209064.

[19] B. S. Baker, D. J. Brown, and H. P. Katseff. A algorithm for two-dimensional packing. *Journal of Algorithms*, 2(4):348–368, Dec. 1981. ISSN 01966774. doi: 10.1016/0196-6774(81)90034-1.

[20] B. Balaji, J. Bell-Masterson, E. Bilgin, A. Damianou, P. M. Garcia, A. Jain, R. Luo, A. Maggiar, B. Narayanaswamy, and C. Ye. ORL: Reinforcement Learning Benchmarks for Online Stochastic Optimization Problems, Dec. 2019.

[21] M. M. Baldi and M. Bruglieri. On the generalized bin packing problem. *International Transactions in Operational Research*, 24(3):425–438, 2017. ISSN 1475-3995. doi: 10.1111/itor.12258.

[22] J. Balogh, J. Békési, G. Dósa, L. Epstein, and A. Levin. Online bin packing with cardinality constraints resolved. *Journal of Computer and System Sciences*, 112:34–49, Sept. 2020. ISSN 0022-0000. doi: 10.1016/j.jcss.2020.03.002.

[23] R. Baltean-Lugojan, P. Bonami, R. Misener, and A. Tramontani. Selecting cutting planes for quadratic semidefinite outer-approximation via trained neural networks. page 37.

[24] N. Bansal and A. Khan. Improved Approximation Algorithm for Two-Dimensional Bin Packing. In *Proceedings of the 2014 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 13–25. Society for Industrial and Applied Mathematics, Dec. 2013. ISBN 978-1-61197-338-9. doi: 10.1137/1.9781611973402.2.

[25] N. Bansal, A. Caprara, and M. Sviridenko. A New Approximation Method for Set Covering Problems, with Applications to Multidimensional Bin Packing. *SIAM J. Comput.*, 39(4):1256–1278, Oct. 2009. ISSN 0097-5397.

[26] J. F. Bard and H. W. Purnomo. Preference scheduling for nurses using column generation. *European Journal of Operational Research*, 164(2):510–534, July 2005. ISSN 0377-2217. doi: 10.1016/j.ejor.2003.06.046.

[27] C. Barut, G. Yildirim, and Y. Tatar. An intelligent and interpretable rule-based metaheuristic approach to task scheduling in cloud systems. *Knowledge-Based Systems*, 284:111241, Jan. 2024. ISSN 0950-7051. doi: 10.1016/j.knosys.2023.111241.

[28] Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of*

Operational Research, 290(2):405–421, Apr. 2021. ISSN 0377-2217. doi: 10.1016/j.ejor.2020.07.063.

[29] B. P. Berg and B. T. Denton. Fast Approximation Methods for Online Scheduling of Outpatient Procedure Centers. *INFORMS Journal on Computing*, 29(4):631–644, Nov. 2017. ISSN 1091-9856. doi: 10.1287/ijoc.2017.0750.

[30] J. Berkey and P. Wang. Two-dimensional finite bin-packing algorithms. *Journal of the Operational Research Society*, 38(5):423–429, 1987. ISSN 0160-5682. doi: 10.1057/jors.1987.70.

[31] S. Berndt, K. Jansen, and K.-M. Klein. Fully dynamic bin packing revisited. *Mathematical Programming*, 179(1):109–155, Jan. 2020. ISSN 1436-4646. doi: 10.1007/s10107-018-1325-x.

[32] D. P. Bertsekas. Approximate dynamic programming, 2008.

[33] D. Bertsimas, V. Gupta, and N. Kallus. Data-driven robust optimization. *Mathematical Programming*, 167(2):235–292, Feb. 2018. ISSN 1436-4646. doi: 10.1007/s10107-017-1125-8.

[34] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, Sept. 2003. ISSN 0360-0300. doi: 10.1145/937503.937505.

[35] S. Bolusani, M. Besançon, K. Bestuzheva, A. Chmiela, J. Dionísio, T. Donkiewicz, J. van Doornmalen, L. Eifler, M. Ghannam, A. Gleixner, C. Graczyk, K. Halbig, I. Hedtke, A. Hoen, C. Hojny, R. van der Hulst, D. Kamp, T. Koch, K. Kofler, J. Lentz, J. Manns, G. Mexi, Erik Mühmer, M. E. Pfetsch, F. Schlösser, F. Serrano, Y. Shinano, M. Turner, S. Vigerske, D. Weninger, and L. Xu. The SCIP optimization suite 9.0. Technical Report, Optimization Online, Feb. 2024.

[36] P. Bonami, A. Lodi, and G. Zarpellon. Learning a Classification of Mixed-Integer Quadratic Programming Problems. In W.-J. van Hoeve, editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, volume 10848, pages 595–604. Springer International Publishing, Cham, 2018. ISBN 978-3-319-93030-5 978-3-319-93031-2. doi: 10.1007/978-3-319-93031-2_43.

[37] J. F. Bonnans. *Convex and Stochastic Optimization*. Universitext. Springer International Publishing, Cham, 2019. ISBN 978-3-030-14976-5 978-3-030-14977-2. doi: 10.1007/978-3-030-14977-2.

[38] J. Boyar, S. Kamali, K. S. Larsen, and A. López-Ortiz. Online Bin Packing with Advice, Dec. 2013.

[39] T. Breitenbach, B. Wilkusz, L. Rasbach, and P. Jahnke. On a method for detecting periods and repeating patterns in time series data with autocorrelation and function approximation. *Pattern Recognition*, 138:109355, June 2023. doi: 10.1016/j.patcog.2023.109355.

[40] E. K. Burke, M. R. Hyde, and G. Kendall. Evolving Bin Packing Heuristics with Genetic Programming. In *Parallel Problem Solving from Nature - Ppsn Ix Springer Lecture Notes in Computer Science. Volume 4193 of Lncs., Reykjavik, Iceland, Springer-Verlag (2006) 860–869*, pages 860–869. Springer, 2006.

[41] E. K. Burke, R. S. R. Hellier, G. Kendall, and G. Whitwell. Complete and robust no-fit polygon generation for the irregular stock cutting problem. *European Journal of Operational Research*, 179(1):27–49, May 2007. ISSN 0377-2217. doi: 10.1016/j.ejor.2006.03.011.

[42] E. K. Burke, M. R. Hyde, and G. Kendall. Providing a memory mechanism to enhance the evolutionary design of heuristics. In *IEEE Congress on*

Evolutionary Computation, pages 1–8, July 2010. doi: 10.1109/CEC.2010.5586388.

[43] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, Dec. 2013. ISSN 0160-5682, 1476-9360. doi: 10.1057/jors.2013.71.

[44] Q. Cai, W. Hang, A. Mirhoseini, G. Tucker, J. Wang, and W. Wei. Reinforcement Learning Driven Heuristic Optimization, June 2019.

[45] R. Calegari, F. S. Fogliatto, F. R. Lucini, M. J. Anzanello, and B. D. Schaan. Surgery scheduling heuristic considering OR downstream and upstream facilities and resources. *BMC Health Services Research*, 20(1):684, July 2020. ISSN 1472-6963. doi: 10.1186/s12913-020-05555-1.

[46] P. Cappanera and M. G. Scutellà. Joint Assignment, Scheduling, and Routing Models to Home Care Optimization: A Pattern-Based Approach. *Transportation Science*, 49(4):830–852, Nov. 2015. ISSN 0041-1655. doi: 10.1287/trsc.2014.0548.

[47] A. Caprara. Packing d-Dimensional Bins in d Stages. *Mathematics of Operations Research*, 33(1):203–215, Feb. 2008. ISSN 0364-765X. doi: 10.1287/moor.1070.0289.

[48] B. Cardoen, E. Demeulemeester, and J. Beliën. Operating room planning and scheduling: A literature review. *European Journal of Operational Research*, 201(3):921–932, Mar. 2010. ISSN 03772217. doi: 10.1016/j.ejor.2009.04.011.

[49] I. Castiñeiras, M. De Cauwer, and B. O'Sullivan. Weibull-Based Benchmarks for Bin Packing. In M. Milano, editor, *Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, pages 207–

222, Berlin, Heidelberg, 2012. Springer. ISBN 978-3-642-33558-7. doi: 10.1007/978-3-642-33558-7_17.

[50] A. Ceselli, Á. Felipe, M. T. Ortuño, G. Righini, and G. Tirado. A Branch-and-Cut-and-Price Algorithm for the Electric Vehicle Routing Problem with Multiple Technologies. *Operations Research Forum*, 2(1):8, Jan. 2021. ISSN 2662-2556. doi: 10.1007/s43069-020-00052-x.

[51] D.-S. Chen, R. G. Batson, and Y. Dang. *Applied Integer Programming: Modeling and Solution*. John Wiley & Sons, Sept. 2011. ISBN 978-1-118-21002-4.

[52] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, May 2017. ISSN 1574-0137. doi: 10.1016/j.cosrev.2016.12.001.

[53] J. Christensen, A. Erera, and D. Pacino. A rolling horizon heuristic for the stochastic cargo mix problem. *Transportation Research Part E: Logistics and Transportation Review*, 123:200–220, Mar. 2019. ISSN 1366-5545. doi: 10.1016/j.tre.2018.10.010.

[54] F. R. K. Chung, M. R. Garey, and D. S. Johnson. On Packing Two-Dimensional Bins. *SIAM Journal on Algebraic Discrete Methods*, 3(1):66–76, Mar. 1982. ISSN 0196-5212, 2168-345X. doi: 10.1137/0603007.

[55] N. M. Cid-Garcia and Y. A. Rios-Solis. Positions and covering: A two-stage methodology to obtain optimal solutions for the 2d-bin packing problem. *PLOS ONE*, 15(4):e0229358, Apr. 2020. ISSN 1932-6203. doi: 10.1371/journal.pone.0229358.

[56] E. Coffman, J. Csirik, D. Johnson, and G. Woeginger. An introduction to bin packing. Apr. 2004.

[57] E. G. Coffman, J. Csirik, G. Galambos, S. Martello, and D. Vigo. Bin Packing Approximation Algorithms: Survey and Classification. In P. M. Pardalos, D.-Z. Du, and R. L. Graham, editors, *Handbook of Combinatorial Optimization*, pages 455–531. Springer New York, New York, NY, 2013. ISBN 978-1-4419-7996-4 978-1-4419-7997-1. doi: 10.1007/978-1-4419-7997-1_35.

[58] E. G. Coffman, Jr., M. R. Garey, D. S. Johnson, and R. E. Tarjan. Performance Bounds for Level-Oriented Two-Dimensional Packing Algorithms. *SIAM Journal on Computing*, 9(4):808–826, Nov. 1980. ISSN 0097-5397, 1095-7111. doi: 10.1137/0209062.

[59] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Dynamic Bin Packing. *SIAM Journal on Computing*, 12(2):227–258, May 1983. ISSN 0097-5397. doi: 10.1137/0212014.

[60] I. Correia, L. Gouveia, and F. Saldanha-da-Gama. Solving the variable size bin packing problem with discretized formulations. *Computers & Operations Research*, 35(6):2103–2113, June 2008. ISSN 0305-0548. doi: 10.1016/j.cor.2006.10.014.

[61] C. Courcoubetis and R. R. Weber. Necessary and Sufficient Conditions for Stability of a Bin-Packing System. *Journal of Applied Probability*, 23(4): 989–999, 1986. ISSN 0021-9002. doi: 10.2307/3214471.

[62] I. I. Cplex. V12. 1: User's manual for CPLEX. *International Business Machines Corporation*, 46(53):157, 2009.

[63] T. G. Crainic, G. Perboli, W. Rei, and R. Tadei. Efficient lower bounds and heuristics for the variable cost and size bin packing problem. *Computers & Operations Research*, 38(11):1474–1482, Nov. 2011. ISSN 0305-0548. doi: 10.1016/j.cor.2011.01.001.

[64] T. G. Crainic, L. Gobbato, G. Perboli, and W. Rei. Logistics capacity planning: A stochastic bin packing formulation and a progressive hedging

meta-heuristic. *European Journal of Operational Research*, 253(2):404–417, Sept. 2016. ISSN 0377-2217. doi: 10.1016/j.ejor.2016.02.040.

[65] J. Csirik. An on-line algorithm for variable-sized bin packing. *Acta Informatica*, 26(8):697–709, Oct. 1989. ISSN 1432-0525. doi: 10.1007/BF00289157.

[66] J. Csirik and D. S. Johnson. Bounded Space On-Line Bin Packing: Best Is Better than First. *Algorithmica*, 31(2):115–138, Oct. 2001. ISSN 1432-0541. doi: 10.1007/s00453-001-0041-7.

[67] J. Csirik, D. S. Johnson, C. Kenyon, J. B. Orlin, P. W. Shor, and R. R. Weber. On the Sum-of-Squares algorithm for bin packing. *J. ACM*, 53(1): 1–65, Jan. 2006. ISSN 0004-5411. doi: 10.1145/1120582.1120583.

[68] R. F. F. da Silva and R. C. S. Schouery. A Branch-and-Cut-and-Price Algorithm for Cutting Stock and Related Problems, Aug. 2023.

[69] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song. Learning Combinatorial Optimization Algorithms over Graphs. *arXiv:1704.01665 [cs, stat]*, Feb. 2018.

[70] G. B. Dantzig and P. Wolfe. Decomposition Principle for Linear Programs. *Operations Research*, 8(1):101–111, 1960. ISSN 0030-364X.

[71] M. De Cauwer, D. Mehta, and B. O'Sullivan. The Temporal Bin Packing Problem: An Application to Workload Management in Data Centres. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 157–164, Nov. 2016. doi: 10.1109/ICTAI.2016.0033.

[72] V. De Maio and D. Kimovski. Multi-objective scheduling of extreme data scientific workflows in Fog. *Future Generation Computer Systems*, 106:171–184, May 2020. ISSN 0167-739X. doi: 10.1016/j.future.2019.12.054.

[73] M. Delorme, M. Iori, and S. Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal*

of Operational Research, 255(1):1–20, Nov. 2016. ISSN 0377-2217. doi: 10.1016/j.ejor.2016.04.030.

[74] J. Desrosiers and M. E. Lübbecke. A Primer in Column Generation. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 1–32. Springer US, Boston, MA, 2005. ISBN 978-0-387-25486-9. doi: 10.1007/0-387-25486-2_1.

[75] C. Dhaenens and L. Jourdan. Metaheuristics for data mining: Survey and opportunities for big data. *Annals of Operations Research*, 314(1):117–140, July 2022. ISSN 1572-9338. doi: 10.1007/s10479-021-04496-0.

[76] D. M. Díaz-López, N. A. López-Valencia, E. M. González-Neira, D. Barrera, D. R. Suárez, M. P. Caro-Gutiérrez, and C. Sefair. A simulation-optimization approach for the surgery scheduling problem: A case study considering stochastic surgical times. *International Journal of Industrial Engineering Computations*, pages 409–422, 2018. ISSN 19232926, 19232934. doi: 10.5267/j.ijiec.2018.1.002.

[77] T. Dokeroglu, T. Kucukyilmaz, and E.-G. Talbi. Hyper-heuristics: A survey and taxonomy. *Computers & Industrial Engineering*, 187:109815, Jan. 2024. ISSN 0360-8352. doi: 10.1016/j.cie.2023.109815.

[78] G. Dósa and Y. He. Bin packing problems with rejection penalties and their dual problems. *Information and Computation*, 204(5):795–815, May 2006. ISSN 0890-5401. doi: 10.1016/j.ic.2006.02.003.

[79] G. Dósa and J. Sgall. First Fit bin packing: A tight analysis. In *DROPS-IDN/v2/Document/10.4230/LIPIcs.STACS.2013.538*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013. doi: 10.4230/LIPIcs.STACS.2013.538.

[80] G. Dósa and J. Sgall. Optimal Analysis of Best Fit Bin Packing. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, editors, *Automata,*

Languages, and Programming, pages 429–441, Berlin, Heidelberg, 2014. Springer. ISBN 978-3-662-43948-7. doi: 10.1007/978-3-662-43948-7_36.

[81] F. Dunke and S. Nickel. A general modeling approach to online optimization with lookahead. *Omega*, 63:134–153, Sept. 2016. ISSN 0305-0483. doi: 10.1016/j.omega.2015.10.009.

[82] L. Epstein. Bin Packing with Rejection Revisited. *Algorithmica*, 56(4): 505–528, Apr. 2010. ISSN 1432-0541. doi: 10.1007/s00453-008-9188-9.

[83] L. Epstein and A. Levin. AFPTAS Results for Common Variants of Bin Packing: A New Method for Handling the Small Items. *SIAM Journal on Optimization*, 20(6):3121–3145, Jan. 2010. ISSN 1052-6234. doi: 10.1137/090767613.

[84] L. Epstein and R. van Stee. This side up! *ACM Trans. Algorithms*, 2(2): 228–243, Apr. 2006. ISSN 1549-6325. doi: 10.1145/1150334.1150339.

[85] S. Erbayrak, V. Özkır, and U. Mahir Yıldırım. Multi-objective 3D bin packing problem with load balance and product family concerns. *Computers & Industrial Engineering*, 159:107518, Sept. 2021. ISSN 0360-8352. doi: 10.1016/j.cie.2021.107518.

[86] S. Erdoğan and E. Miller-Hooks. A Green Vehicle Routing Problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):100–114, Jan. 2012. ISSN 1366-5545. doi: 10.1016/j.tre.2011.08.001.

[87] S. A. Erdogan and B. T. Denton. *Surgery Planning and Scheduling*. Wiley, 1 edition, Jan. 2011. ISBN 978-0-470-40063-0 978-0-470-40053-1. doi: 10.1002/9780470400531.eorms0861.

[88] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37–37, Mar. 1996. ISSN 2371-9621. doi: 10.1609/aimag.v17i3.1230.

[89] D. Feng, Z. Wu, D. Zuo, and Z. Zhang. A multiobjective migration algorithm as a resource consolidation strategy in cloud computing. *PLOS ONE*, 14(2): e0211729, Feb. 2019. ISSN 1932-6203. doi: 10.1371/journal.pone.0211729.

[90] A. Fernández, C. Gil, R. Baños, and M. G. Montoya. A parallel multi-objective algorithm for two-dimensional bin packing with rotations and load balancing. *Expert Systems with Applications*, 40(13):5169–5180, Oct. 2013. ISSN 0957-4174. doi: 10.1016/j.eswa.2013.03.015.

[91] R. B. Fetter, Y. Shin, J. L. Freeman, R. F. Averill, and J. D. Thompson. Case mix definition by diagnosis-related groups. *Medical Care*, 18(2 Suppl): iii, 1–53, Feb. 1980. ISSN 0025-7079.

[92] J. B. G. Frenk and G. Galambos. Hybrid next-fit algorithm for the two-dimensional rectangle bin-packing problem. *Computing*, 39(3):201–217, Sept. 1987. ISSN 0010-485X, 1436-5057. doi: 10.1007/BF02309555.

[93] Fuce and Wanghui. The solving strategy for the real-world vehicle routing problem. In *2010 3rd International Congress on Image and Signal Processing*, pages 3182–3185, Yantai, Oct. 2010. IEEE. ISBN 978-1-4244-6513-2 978-1-4244-6516-3. doi: 10.1109/CISP.2010.5647968.

[94] G. Galambos and G. J. Woeginger. On-line bin packing — A restricted survey. *Zeitschrift für Operations Research*, 42(1):25–45, 1995. ISSN 1432-5217. doi: 10.1007/BF01415672.

[95] W. Gálvez, F. Grandoni, S. Ingala, and A. Khan. Improved Pseudo-Polynomial-Time Approximation for Strip Packing, Jan. 2018.

[96] G. Gambosi, A. Postiglione, and M. Talamo. Algorithms for the Relaxed Online Bin-Packing Model. *SIAM Journal on Computing*, July 2006. doi: 10.1137/S0097539799180408.

[97] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1979. ISBN 978-0-7167-1044-8.

[98] M. R. Garey, R. L. Graham, D. S. Johnson, and A. C.-C. Yao. Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory, Series A*, 21(3):257–298, Nov. 1976. ISSN 0097-3165. doi: 10.1016/0097-3165(76)90001-7.

[99] M. Gasse, D. Chetelat, N. Ferroni, L. Charlin, and A. Lodi. Exact Combinatorial Optimization with Graph Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[100] B. Gendron, P.-V. Khuong, and F. Semet. A Lagrangian-Based Branch-and-Bound Algorithm for the Two-Level Uncapacitated Facility Location Problem with Single-Assignment Constraints. *Transportation Science*, 50 (4):1286–1299, Nov. 2016. ISSN 0041-1655, 1526-5447. doi: 10.1287/trsc.2016.0692.

[101] P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting Stock Problem—Part II. *Operations Research*, 11(6):863–888, Dec. 1963. ISSN 0030-364X, 1526-5463. doi: 10.1287/opre.11.6.863.

[102] L. Glomb, F. Liers, and F. Rösel. A rolling-horizon approach for multi-period optimization. *European Journal of Operational Research*, 300(1): 189–206, July 2022. ISSN 0377-2217. doi: 10.1016/j.ejor.2021.07.043.

[103] F. Glover and K. Sorensen. Metaheuristics. *Scholarpedia*, 10(4):6532, 2015. ISSN 1941-6016. doi: 10.4249/scholarpedia.6532.

[104] I. Golan. Performance Bounds for Orthogonal Oriented Two-Dimensional Packing Algorithms. *SIAM Journal on Computing*, 10(3):571–582, Aug. 1981. ISSN 0097-5397. doi: 10.1137/0210042.

[105] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, Sept. 1958. ISSN 0002-9904, 1936-881X.

[106] B. GRABOT and L. GENESTE. Dispatching rules in scheduling Dispatching rules in scheduling: A fuzzy approach. *International Journal of Production Research*, 32(4):903–915, Apr. 1994. ISSN 0020-7543. doi: 10.1080/00207549408956978.

[107] V. Griffiths, J. P. Scanlan, M. H. Eres, A. Martinez-Sykora, and P. Chinchapatnam. Cost-driven build orientation and bin packing of parts in Selective Laser Melting (SLM). *European Journal of Operational Research*, 273(1): 334–352, Feb. 2019. ISSN 0377-2217. doi: 10.1016/j.ejor.2018.07.053.

[108] E. F. Grove. Online bin packing with lookahead. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '95, pages 430–436, USA, Jan. 1995. Society for Industrial and Applied Mathematics. ISBN 978-0-89871-349-7.

[109] S. Gu, R. Cheng, and Y. Jin. Feature selection for high-dimensional classification using a competitive swarm optimizer. *Soft Computing*, 22(3):811–822, Feb. 2018. ISSN 1433-7479. doi: 10.1007/s00500-016-2385-6.

[110] R. Guido and D. Conforti. A hybrid genetic approach for solving an integrated multi-objective operating room planning and scheduling problem. *Computers & Operations Research*, 87:270–282, Nov. 2017. ISSN 0305-0548. doi: 10.1016/j.cor.2016.11.009.

[111] S. Gul. A Stochastic Programming Approach for Appointment Scheduling Under Limited Availability of Surgery Turnover Teams. *Service Science*, 10 (3):277–288, Sept. 2018. ISSN 2164-3962. doi: 10.1287/serv.2018.0214.

[112] S. Gul, B. T. Denton, and J. W. Fowler. A Progressive Hedging Approach

for Surgery Planning Under Uncertainty. *INFORMS Journal on Computing*, 27(4):755–772, Nov. 2015. ISSN 1091-9856. doi: 10.1287/ijoc.2015.0658.

[113] B. Guo, Y. Zhang, J. Hu, J. Li, F. Wu, Q. Peng, and Q. Zhang. Two-dimensional irregular packing problems: A review. *Frontiers in Mechanical Engineering*, 8, 2022. ISSN 2297-3079.

[114] V. Gupta and A. Radovanovic. Online Stochastic Bin Packing. *Operations Research*, 68(5):1474–1492, Sept. 2020. ISSN 0030-364X, 1526-5463. doi: 10.1287/opre.2019.1914.

[115] Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2024.

[116] G. Haffari and S. B. Shouraki. Analysis of STAGE Algorithm based on Solving Bin Packing Problem. page 7.

[117] R. Hall, editor. *Handbook of Healthcare System Scheduling*, volume 168 of *International Series in Operations Research & Management Science*. Springer US, Boston, MA, 2012. ISBN 978-1-4614-1733-0 978-1-4614-1734-7. doi: 10.1007/978-1-4614-1734-7.

[118] X. Han, F. Y. L. Chin, H.-F. Ting, G. Zhang, and Y. Zhang. A new upper bound 2.5545 on 2D Online Bin Packing. *ACM Trans. Algorithms*, 7(4): 50:1–50:18, Sept. 2011. ISSN 1549-6325. doi: 10.1145/2000807.2000818.

[119] X. Han, K. Iwama, D. Ye, and G. Zhang. Approximate strip packing: Revisited. *Information and Computation*, 249:110–120, Aug. 2016. ISSN 0890-5401. doi: 10.1016/j.ic.2016.03.010.

[120] N. Hansen, D. V. Arnold, and A. Auger. Evolution Strategies. In J. Kacprzyk and W. Pedrycz, editors, *Springer Handbook of Computational Intelligence*, pages 871–898. Springer, Berlin, Heidelberg, 2015. ISBN 978-3-662-43505-2. doi: 10.1007/978-3-662-43505-2_44.

[121] M. Haouari and M. Serairi. Heuristics for the variable sized bin-packing problem. *Computers & Operations Research*, 36(10):2877–2884, Oct. 2009. ISSN 0305-0548. doi: 10.1016/j.cor.2008.12.016.

[122] S. Harris and D. Claudio. Current Trends in Operating Room Scheduling 2015 to 2020: A Literature Review. *Operations Research Forum*, 3(1):21, Mar. 2022. ISSN 2662-2556. doi: 10.1007/s43069-022-00134-y.

[123] K. Helsgaun. An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems. page 60.

[124] V. Hemmelmayr, V. Schmid, and C. Blum. Variable neighbourhood search for the variable sized bin packing problem. *Computers & Operations Research*, 39(5):1097–1108, May 2012. ISSN 0305-0548. doi: 10.1016/j.cor.2011.07.003.

[125] J. H. Holland. Genetic Algorithms. *Scientific American*, 267(1):66–73, 1992. ISSN 0036-8733.

[126] R. A. Howard. Dynamic programming and markov processes. 1960.

[127] H. Hu, X. Zhang, X. Yan, L. Wang, and Y. Xu. Solving a New 3D Bin Packing Problem with Deep Reinforcement Learning Method, Aug. 2017.

[128] Y. Huang, L. Lai, W. Li, and H. Wang. A differential evolution algorithm with ternary search tree for solving the three-dimensional packing problem. *Information Sciences*, 606:440–452, Aug. 2022. ISSN 0020-0255. doi: 10.1016/j.ins.2022.05.063.

[129] Z. Huang, K. Wang, F. Liu, H.-L. Zhen, W. Zhang, M. Yuan, J. Hao, Y. Yu, and J. Wang. Learning to select cuts for efficient mixed-integer programming. *Pattern Recognition*, 123:108353, Mar. 2022. ISSN 0031-3203. doi: 10.1016/j.patcog.2021.108353.

[130] Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, Mar. 2018. ISSN 1867-2949, 1867-2957. doi: 10.1007/s12532-017-0130-5.

[131] W. Irwin-Harris, Y. Sun, B. Xue, and M. Zhang. A Graph-Based Encoding for Evolutionary Convolutional Neural Network Architecture Design. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 546–553, June 2019. doi: 10.1109/CEC.2019.8790093.

[132] G. Ismayilov and H. R. Topcuoglu. Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing. *Future Generation Computer Systems*, 102:307–322, Jan. 2020. ISSN 0167-739X. doi: 10.1016/j.future.2019.08.012.

[133] Z. Ivkovic and E. L. Lloyd. Fully Dynamic Algorithms for Bin Packing: Being (Mostly) Myopic Helps. *SIAM Journal on Computing*, 28(2):574–611, Jan. 1998. ISSN 0097-5397. doi: 10.1137/S0097539794276749.

[134] J. Jacques, H. Martin-Huyghe, J. Lemtiri-Florek, J. Taillard, L. Jourdan, C. Dhaenens, D. Delerue, A. Hansske, and V. Leclercq. The detection of hospitalized patients at risk of testing positive to multi-drug resistant bacteria using MOCA-I, a rule-based "white-box" classification algorithm for medical data. *International Journal of Medical Informatics*, 142:104242, Oct. 2020. ISSN 1386-5056. doi: 10.1016/j.ijmedinf.2020.104242.

[135] A. Jain, R. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1): 4–37, Jan. 2000. ISSN 1939-3539. doi: 10.1109/34.824819.

[136] C. Jin. 0-1 Knapsack in Nearly Quadratic Time. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 271–282, Vancouver BC Canada, June 2024. ACM. ISBN 9798400703836. doi: 10.1145/3618260.3649618.

[137] M. Jin, H. Y. Koh, Q. Wen, D. Zambon, C. Alippi, G. I. Webb, I. King, and S. Pan. A Survey on Graph Neural Networks for Time Series: Forecasting, Classification, Imputation, and Anomaly Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10466–10485, Dec. 2024. ISSN 1939-3539. doi: 10.1109/TPAMI.2024.3443141.

[138] S. R. Jodogne and J. H. Piater. Closed-Loop Learning of Visual Control Policies. *Journal of Artificial Intelligence Research*, 28:349–391, Mar. 2007. ISSN 1076-9757. doi: 10.1613/jair.2110.

[139] D. S. Johnson. *Near-Optimal Bin Packing Algorithms*. PhD thesis, Massachusetts Institute of Technology, 1973.

[140] D. S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8(3):272–314, June 1974. ISSN 0022-0000. doi: 10.1016/S0022-0000(74)80026-7.

[141] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms. *SIAM Journal on Computing*, July 2006. doi: 10.1137/0203025.

[142] S. Kagaya, S. Kikuchi, and R. A. Donnelly. Use of a fuzzy theory technique for grouping of trips in the vehicle routing and scheduling problem. *European Journal of Operational Research*, 76(1):143–154, July 1994. ISSN 0377-2217. doi: 10.1016/0377-2217(94)90012-4.

[143] J. Kang and S. Park. Algorithms for the variable sized bin packing problem. *European Journal of Operational Research*, 147(2):365–372, June 2003. ISSN 0377-2217. doi: 10.1016/S0377-2217(02)00247-3.

[144] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science (Sfcs 1982)*, pages 312–320, Nov. 1982. doi: 10.1109/SFCS.1982.61.

[145] N. Kashyap and A. Mishra. Chapter 1 - A discourse on metaheuristics techniques for solving clustering and semisupervised learning models. In S. Mishra, H. K. Tripathy, P. K. Mallick, A. K. Sangaiah, and G.-S. Chae, editors, *Cognitive Big Data Intelligence with a Metaheuristic Approach*, Cognitive Data Science in Sustainable Computing, pages 1–19. Academic Press, Jan. 2022. ISBN 978-0-323-85117-6. doi: 10.1016/B978-0-323-85117-6.00012-1.

[146] H. Kellerer and U. Pferschy. Cardinality constrained bin-packing problems. *Annals of Operations Research*, 92(0):335–348, Jan. 1999. ISSN 1572-9338. doi: 10.1023/A:1018947117526.

[147] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948, Perth, WA, Australia, 1995. IEEE. ISBN 978-0-7803-2768-9. doi: 10.1109/ICNN.1995.488968.

[148] E. Khalil, P. L. Bodic, L. Song, G. Nemhauser, and B. Dilkina. Learning to Branch in Mixed Integer Programming. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Feb. 2016. ISSN 2374-3468. doi: 10.1609/aaai.v30i1.10080.

[149] W. Kool, H. van Hoof, and M. Welling. Attention, Learn to Solve Routing Problems! In *7th International Conference on Learning Representations, ICLR 2019*. International Conference on Learning Representations, ICLR, 2019.

[150] D. A. Koonce and S. C. Tsai. Using data mining to find patterns in genetic algorithm solutions to a job shop schedule. *Computers & Industrial Engineering*, 38(3):361–374, Oct. 2000. ISSN 0360-8352. doi: 10.1016/S0360-8352(00)00050-4.

[151] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algo-*

rithms, volume 21 of *Algorithms and Combinatorics*. Springer, Berlin, Heidelberg, 2012. ISBN 978-3-642-24487-2 978-3-642-24488-9. doi: 10.1007/978-3-642-24488-9.

[152] J. R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, June 1994. ISSN 1573-1375. doi: 10.1007/BF00175355.

[153] K. L. Krause, V. Y. Shen, and H. D. Schwetman. Analysis of Several Task-Scheduling Algorithms for a Model of Multiprogramming Computer Systems. *J. ACM*, 22(4):522–550, Oct. 1975. ISSN 0004-5411. doi: 10.1145/321906.321917.

[154] M. Kruber, M. E. Lübbecke, and A. Parmentier. Learning When to Use a Decomposition. In D. Salvagnin and M. Lombardi, editors, *Integration of AI and OR Techniques in Constraint Programming*, volume 10335, pages 202–210. Springer International Publishing, Cham, 2017. ISBN 978-3-319-59775-1 978-3-319-59776-8. doi: 10.1007/978-3-319-59776-8_16.

[155] T. Kucukyilmaz and H. E. Kiziloz. Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem. *Computers & Industrial Engineering*, 125:157–170, Nov. 2018. ISSN 0360-8352. doi: 10.1016/j.cie.2018.08.021.

[156] O. Kundu, S. Dutta, and S. Kumar. Deep-Pack: A Vision-Based 2D Online Bin Packing Algorithm with Deep Reinforcement Learning. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1–7, Oct. 2019. doi: 10.1109/RO-MAN46459.2019.8956393.

[157] T. Kuosmanen and X. Zhou. Shadow prices and marginal abatement costs: Convex quantile regression approach. *European Journal of Op-*

erational Research, 289(2):666–675, Mar. 2021. ISSN 0377-2217. doi: 10.1016/j.ejor.2020.07.036.

[158] E. Larsen, S. Lachapelle, Y. Bengio, E. Frejinger, S. Lacoste-Julien, and A. Lodi. Predicting Tactical Solutions to Operational Planning Problems under Imperfect Information. *arXiv:1807.11876 [cs, stat]*, Mar. 2021.

[159] A. Laterre, Y. Fu, M. K. Jabri, A.-S. Cohen, D. Kas, K. Hajjar, T. S. Dahl, A. Kerkeni, and K. Beguir. Ranked Reward: Enabling Self-Play Reinforcement Learning for Combinatorial Optimization, Dec. 2018.

[160] E. L. Lawler and D. E. Wood. Branch-and-Bound Methods: A Survey. *Operations Research*, 14(4):699–719, Aug. 1966. ISSN 0030-364X, 1526-5463. doi: 10.1287/opre.14.4.699.

[161] A. A. S. Leao, F. M. B. Toledo, J. F. Oliveira, M. A. Carravilla, and R. Alvarez-Valdés. Irregular packing problems: A review of mathematical models. *European Journal of Operational Research*, 282(3):803–822, May 2020. ISSN 0377-2217. doi: 10.1016/j.ejor.2019.04.045.

[162] C. C. Lee and D. T. Lee. A simple on-line bin-packing algorithm. *Journal of the ACM*, 32(3):562–572, July 1985. ISSN 0004-5411, 1557-735X. doi: 10.1145/3828.3833.

[163] G. Leeftink and E. W. Hans. Case mix classification and a benchmark set for surgery scheduling. *Journal of Scheduling*, 21(1):17–33, Feb. 2018. ISSN 1099-1425. doi: 10.1007/s10951-017-0539-8.

[164] J. Y.-T. Leung, T. W. Tam, C. S. Wong, G. H. Young, and F. Y. L. Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275, Nov. 1990. ISSN 0743-7315. doi: 10.1016/0743-7315(90)90019-L.

[165] J. Levine and F. Ducatelle. Ant Colony Optimisation and Local Search for Bin Packing and Cutting Stock Problems.

[166] D. Li, Z. Gu, Y. Wang, C. Ren, and F. C. Lau. One model packs thousands of items with Recurrent Conditional Query Learning. *Knowledge-Based Systems*, 235:107683, Jan. 2022. ISSN 09507051. doi: 10.1016/j.knosys.2021.107683.

[167] Y. Li. Deep Reinforcement Learning: An Overview. *arXiv:1701.07274 [cs]*, Nov. 2018.

[168] Y. Li, X. Tang, and W. Cai. Dynamic Bin Packing for On-Demand Cloud Resource Allocation. *IEEE Transactions on Parallel and Distributed Systems*, 27(1):157–170, Jan. 2016. ISSN 1558-2183. doi: 10.1109/TPDS.2015.2393868.

[169] Y. Li, J.-P. Clarke, and S. S. Dey. Using submodularity within column generation to solve the flight-to-gate assignment problem. *Transportation Research Part C: Emerging Technologies*, 129:103217, Aug. 2021. ISSN 0968-090X. doi: 10.1016/j.trc.2021.103217.

[170] B. Lin, J. Li, R. Bai, R. Qu, T. Cui, and H. Jin. Identify Patterns in Online Bin Packing Problem: An Adaptive Pattern-Based Algorithm. *Symmetry*, 14(7):1301, July 2022. ISSN 2073-8994. doi: 10.3390/sym14071301.

[171] B. Lin, J. Li, T. Cui, H. Jin, R. Bai, R. Qu, and J. Garibaldi. A pattern-based algorithm with fuzzy logic bin selector for online bin packing problem. *Expert Systems with Applications*, 249:123515, Sept. 2024. ISSN 0957-4174. doi: 10.1016/j.eswa.2024.123515.

[172] Z. Lin, X. Jiang, and Z. Zheng. A coarse-to-fine pattern parser for mitigating the issue of drastic imbalance in pixel distribution. *Pattern Recognition*, 148: 110143, Apr. 2024. doi: 10.1016/j.patcog.2023.110143.

[173] D. S. Liu, K. C. Tan, S. Y. Huang, C. K. Goh, and W. K. Ho. On solving multiobjective bin packing problems using evolutionary particle swarm op-

timization. *European Journal of Operational Research*, 190(2):357–382, Oct. 2008. ISSN 0377-2217. doi: 10.1016/j.ejor.2007.06.032.

[174] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan. A Survey on Evolutionary Neural Architecture Search. *IEEE Transactions on Neural Networks and Learning Systems*, 34(2):550–570, Feb. 2023. ISSN 2162-2388. doi: 10.1109/TNNLS.2021.3100554.

[175] A. Lodi, S. Martello, and D. Vigo. Approximation algorithms for the oriented two-dimensional bin packing problem. *European Journal of Operational Research*, 112(1):158–166, Jan. 1999. ISSN 03772217. doi: 10.1016/S0377-2217(97)00388-3.

[176] A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252, Sept. 2002. ISSN 0377-2217. doi: 10.1016/S0377-2217(02)00123-6.

[177] E. López-Camacho, H. Terashima-Marin, P. Ross, and G. Ochoa. A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications*, 41(15):6876–6889, Nov. 2014. ISSN 0957-4174. doi: 10.1016/j.eswa.2014.04.043.

[178] M. E. Lübbecke and J. Desrosiers. Selected Topics in Column Generation. *Operations Research*, 53(6):1007–1023, Dec. 2005. ISSN 0030-364X. doi: 10.1287/opre.1050.0234.

[179] G. Ma and E. Demeulemeester. A multilevel integrative approach to hospital case mix and capacity planning. *Computers & Operations Research*, 40(9): 2198–2207, Sept. 2013. ISSN 0305-0548. doi: 10.1016/j.cor.2012.01.013.

[180] A. Macario, T. S. Vitez, B. Dunn, and T. McDonald. Where are the costs in perioperative care? Analysis of hospital costs and charges for inpatient surgical care. *Anesthesiology*, 83(6):1138–1144, Dec. 1995. ISSN 0003-3022. doi: 10.1097/00000542-199512000-00002.

[181] W. Mao. Besk-k-Fit bin packing. *Computing*, 50(3):265–270, Sept. 1993. ISSN 1436-5057. doi: 10.1007/BF02243816.

[182] F. Marinelli, A. Pizzuti, W. Wu, and M. Yagiura. One-dimensional bin packing with pattern-dependent processing time. *European Journal of Operational Research*, Nov. 2024. ISSN 0377-2217. doi: 10.1016/j.ejor.2024.11.023.

[183] H. Marković, I. Ćavar, and T. Carić. Using Data Mining to Forecast Uncertain Demands in Stochastic Vehicle Routing Problem. In H. Boštjan, M. Anžek, and S. Janša, editors, *Zbornik Referatov = Proceedings ISEP 2005*, page U5. Ljubljana: Elektrotehniška zveza Slovenije, 2005. ISBN 978-961-6187-34-3.

[184] S. Martello. Knapsack problems: Algorithms and computer implementations. *Wiley-Interscience series in discrete mathematics and optimiza tion*, 1990.

[185] S. Martello and P. Toth. Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics*, 28(1):59–70, July 1990. ISSN 0166-218X. doi: 10.1016/0166-218X(90)90094-S.

[186] S. Martello, D. Pisinger, and D. Vigo. The Three-Dimensional Bin Packing Problem. *Operations Research*, 48(2):256–267, 2000. ISSN 0030-364X. doi: 10.1287/opre.48.2.256.12386.

[187] J. Martinovic and M. Selch. Mathematical models and approximate solution approaches for the stochastic bin packing problem. *Computers & Operations Research*, 135:105439, Nov. 2021. ISSN 0305-0548. doi: 10.1016/j.cor.2021.105439.

[188] M. L. McManus, M. C. Long, A. Cooper, J. Mandell, D. M. Berwick, M. Pagano, and E. Litvak. Variability in surgical caseload and access to intensive care services. *Anesthesiology*, 98(6):1491–1496, June 2003. ISSN 0003-3022. doi: 10.1097/00000542-200306000-00029.

[189] F. Meng, B. Cao, D. Chu, Q. Ji, and X. Zhou. Variable neighborhood search for quadratic multiple constraint variable sized bin-packing problem. *Computers & Operations Research*, 143:105803, July 2022. ISSN 0305-0548. doi: 10.1016/j.cor.2022.105803.

[190] R. M'Hallah and A. Bouziri. Heuristics for the combined cut order planning two-dimensional layout problem in the apparel industry. *International Transactions in Operational Research*, 23(1-2):321–353, 2016. ISSN 1475-3995. doi: 10.1111/itor.12104.

[191] S. Mirjalili. Ant Colony Optimisation. In S. Mirjalili, editor, *Evolutionary Algorithms and Neural Networks: Theory and Applications*, pages 33–42. Springer International Publishing, Cham, 2019. ISBN 978-3-319-93025-1. doi: 10.1007/978-3-319-93025-1_3.

[192] T. M. Mitchell. *Machine Learning*. McGraw-Hill Series in Computer Science. McGraw-Hill, New York, 1997. ISBN 978-0-07-042807-2.

[193] H. D. Mittelmann. Latest benchmarks of optimization software. In *INFORMS Annual Meeting. Houston, TX*, 2017.

[194] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *arXiv:1602.01783 [cs]*, June 2016.

[195] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, Feb. 2016. ISSN 1572-5286. doi: 10.1016/j.disopt.2016.01.005.

[196] N. Nezamoddini and S. S. Lam. Reliability and topology based network design using pattern mining guided genetic algorithm. *Expert Systems with Applications*, 42(21):7483–7492, Nov. 2015. ISSN 0957-4174. doi: 10.1016/j.eswa.2015.05.019.

[197] R. Nian, J. Liu, and B. Huang. A review On reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, 139:106886, Aug. 2020. ISSN 00981354. doi: 10.1016/j.compchemeng.2020.106886.

[198] L. Nieddu and G. Patrizi. Formal methods in pattern recognition: A review. *European Journal of Operational Research*, 120(3):459–495, Feb. 2000. ISSN 0377-2217. doi: 10.1016/S0377-2217(98)00368-3.

[199] P. Norvig and S. Russell. *Artificial Intelligence: A Modern Approach, Global Edition*. Pearson, Boston, 4th edition edition, May 2021. ISBN 978-1-292-40113-3.

[200] J. Nyman and K. S. N. Ripon. Metaheuristics for the Multiobjective Surgery Admission Planning Problem. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, July 2018. doi: 10.1109/CEC.2018.8477791.

[201] B. Pang, X. Xie, Y. Song, and L. Luo. Surgery Scheduling Under Case Cancellation and Surgery Duration Uncertainty. *IEEE Transactions on Automation Science and Engineering*, 16(1):74–86, Jan. 2019. ISSN 1558-3783. doi: 10.1109/TASE.2018.2834486.

[202] B. Peng, J. Wang, and Z. Zhang. A Deep Reinforcement Learning Algorithm Using Dynamic Attention Model for Vehicle Routing Problems. In K. Li, W. Li, H. Wang, and Y. Liu, editors, *Artificial Intelligence Algorithms and Applications*, Communications in Computer and Information Science, pages 636–650, Singapore, 2020. Springer. ISBN 9789811555770. doi: 10.1007/978-981-15-5577-0_51.

[203] G. Perboli, R. Tadei, and M. M. Baldi. The stochastic generalized bin packing problem. *Discrete Applied Mathematics*, 160(7):1291–1297, May 2012. ISSN 0166-218X. doi: 10.1016/j.dam.2011.10.037.

[204] L. Perron and F. Didier. CP-SAT. Google, May 2024.

[205] D. T. Phan. Lagrangian Duality and Branch-and-Bound Algorithms for Optimal Power Flow. *Operations Research*, 60(2):275–285, 2012. ISSN 0030-364X.

[206] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, Aug. 2007. ISSN 0305-0548. doi: 10.1016/j.cor.2005.09.012.

[207] D. Pisinger and S. Ropke. Large Neighborhood Search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 99–127. Springer International Publishing, Cham, 2019. ISBN 978-3-319-91086-4. doi: 10.1007/978-3-319-91086-4_4.

[208] H. Pourvaziri, H. Sarhadi, N. Azad, H. Afshari, and M. Taghavi. Planning of electric vehicle charging stations: An integrated deep learning and queueing theory approach. *Transportation Research Part E: Logistics and Transportation Review*, 186:103568, June 2024. ISSN 1366-5545. doi: 10.1016/j.tre.2024.103568.

[209] R. Qian, X. Lai, and X. Li. 3D Object Detection for Autonomous Driving: A Survey. *Pattern Recognition*, 130:108796, Oct. 2022. ISSN 0031-3203. doi: 10.1016/j.patcog.2022.108796.

[210] Y. Qin, F. T. S. Chan, S. H. Chung, T. Qu, and B. Niu. Aircraft parking stand allocation problem with safety consideration for independent hangar maintenance service providers. *Computers & Operations Research*, 91:225–236, Mar. 2018. ISSN 0305-0548. doi: 10.1016/j.cor.2017.10.001.

[211] S. Rachuba and B. Werners. A fuzzy multi-criteria approach for robust operating room schedules. *Annals of Operations Research*, 251(1):325–350, Apr. 2017. ISSN 1572-9338. doi: 10.1007/s10479-015-1926-1.

[212] G. R. Raidl. A Unified View on Hybrid Metaheuristics. In F. Almeida, M. J. Blesa Aguilera, C. Blum, J. M. Moreno Vega, M. Pérez Pérez, A. Roli, and

M. Sampels, editors, *Hybrid Metaheuristics*, pages 1–12, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-46385-6. doi: 10.1007/11890584_1.

[213] R. Ren, X. Tang, Y. Li, and W. Cai. Competitiveness of Dynamic Bin Packing for Online Cloud Server Allocation. *IEEE/ACM Transactions on Networking*, 25(3):1324–1331, June 2017. ISSN 1558-2566. doi: 10.1109/TNET.2016.2630052.

[214] G. d. A. Rodrigues, C. B. Cunha, L. Guarino Neto, and J. G. V. Vieira. An adaptive large neighborhood search metaheuristic for the Generalized Bin Packing problem with incompatible categories. *Computers & Industrial Engineering*, 185:109586, Nov. 2023. ISSN 0360-8352. doi: 10.1016/j.cie.2023.109586.

[215] B. Romera-Paredes, M. Barekatain, A. Novikov, M. Balog, M. P. Kumar, E. Dupont, F. J. R. Ruiz, J. S. Ellenberg, P. Wang, O. Fawzi, P. Kohli, and A. Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, Jan. 2024. ISSN 1476-4687. doi: 10.1038/s41586-023-06924-6.

[216] L.-M. Rousseau, M. Gendreau, G. Pesant, and F. Focacci. Solving VRPTWs with Constraint Programming Based Column Generation. *Annals of Operations Research*, 130(1):199–216, Aug. 2004. ISSN 1572-9338. doi: 10.1023/B:ANOR.0000032576.73681.29.

[217] T. P. Runarsson and A. O. Sigurpalsson. Towards an evolutionary guided exact solution to elective surgery scheduling under uncertainty and ward restrictions. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 419–425, June 2019. doi: 10.1109/CEC.2019.8790174.

[218] S. J. Russell, P. Norvig, and E. Davis. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, Upper Saddle River, 3rd ed edition, 2010. ISBN 978-0-13-604259-4.

[219] Ö. Şafak and G. Erdoğan. A Large Neighbourhood Search Algorithm for Solving Container Loading Problems. *Computers & Operations Research*, 154:106199, June 2023. ISSN 0305-0548. doi: 10.1016/j.cor.2023.106199.

[220] G. Sagnol, C. Barner, R. Borndörfer, M. Grima, M. Seeling, C. Spies, and K. Wernecke. Robust allocation of operating rooms: A cutting plane approach to handle lognormal case durations. *European Journal of Operational Research*, 271(2):420–435, Dec. 2018. ISSN 0377-2217. doi: 10.1016/j.ejor.2018.05.022.

[221] M. Samudra, C. Van Riet, E. Demeulemeester, B. Cardoen, N. Vansteenkiste, and F. E. Rademakers. Scheduling operating rooms: Achievements, challenges and pitfalls. *Journal of Scheduling*, 19(5):493–525, Oct. 2016. ISSN 1094-6136, 1099-1425. doi: 10.1007/s10951-016-0489-6.

[222] L. Scavuzzo, K. Aardal, A. Lodi, and N. Yorke-Smith. Machine learning augmented branch and bound for mixed integer linear programming. *Mathematical Programming*, Aug. 2024. ISSN 1436-4646. doi: 10.1007/s10107-024-02130-y.

[223] G. Scheithauer. *Introduction to Cutting and Packing Optimization*, volume 263 of *International Series in Operations Research & Management Science*. Springer International Publishing, Cham, 2018. ISBN 978-3-319-64402-8 978-3-319-64403-5. doi: 10.1007/978-3-319-64403-5.

[224] J. Schmidhuber. Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61:85–117, Aug. 2014. ISSN 08936080. doi: 10.1016/j.neunet.2014.09.003.

[225] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*, Aug. 2017.

[226] S. S. Seiden. On the online bin packing problem. *Journal of the ACM*, 49 (5):640–671, Sept. 2002. ISSN 0004-5411. doi: 10.1145/585265.585269.

[227] J. Sheng, Y. Hu, W. Zhou, L. Zhu, B. Jin, J. Wang, and X. Wang. Learning to schedule multi-NUMA virtual machines via reinforcement learning. *Pattern Recognition*, 121:108254, Jan. 2022. ISSN 0031-3203. doi: 10.1016/j.patcog.2021.108254.

[228] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, Jan. 2016. doi: 10.1038/nature16961.

[229] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv:1712.01815 [cs]*, Dec. 2017.

[230] R. M. Singh, L. K. Awasthi, and G. Sikka. Towards Metaheuristic Scheduling Techniques in Cloud and Fog: An Extensive Taxonomic Review. *ACM Comput. Surv.*, 55(3):50:1–50:43, Feb. 2022. ISSN 0360-0300. doi: 10.1145/3494520.

[231] K. Sörensen. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18, 2015. ISSN 1475-3995. doi: 10.1111/itor.12001.

[232] A. Stawowy. Evolutionary based heuristic for bin packing problem. *Computers & Industrial Engineering*, 55(2):465–474, Sept. 2008. ISSN 0360-8352. doi: 10.1016/j.cie.2008.01.007.

[233] P. S. Stepaniak, C. Heij, and G. De Vries. Modeling and prediction of surgical procedure times. *Statistica Neerlandica*, 64(1):1–18, 2010. ISSN 1467-9574. doi: 10.1111/j.1467-9574.2009.00440.x.

[234] P. Stokkink, J.-F. Cordeau, and N. Geroliminis. A column and row generation approach to the crowd-shipping problem with transfers. *Omega*, 128: 103134, Oct. 2024. ISSN 0305-0483. doi: 10.1016/j.omega.2024.103134.

[235] Y. Sun, F. Lin, and H. Xu. Multi-objective Optimization of Resource Scheduling in Fog Computing Using an Improved NSGA-II. *Wireless Personal Communications*, 102(2):1369–1385, Sept. 2018. ISSN 1572-834X. doi: 10.1007/s11277-017-5200-5.

[236] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass, 1998. ISBN 978-0-262-19398-6.

[237] R. E. Tarjan. Amortized Computational Complexity. *SIAM Journal on Algebraic Discrete Methods*, 6(2):306–318, Apr. 1985. ISSN 0196-5212. doi: 10.1137/0606031.

[238] Y. Tian, C. Lu, X. Zhang, F. Cheng, and Y. Jin. A Pattern Mining-Based Evolutionary Algorithm for Large-Scale Sparse Multiobjective Optimization Problems. *IEEE Transactions on Cybernetics*, 52(7):6784–6797, July 2022. ISSN 2168-2275. doi: 10.1109/TCYB.2020.3041325.

[239] K. Tole, R. Moqa, J. Zheng, and K. He. A Simulated Annealing approach for the Circle Bin Packing Problem with Rectangular Items. *Computers & Industrial Engineering*, 176:109004, Feb. 2023. ISSN 0360-8352. doi: 10.1016/j.cie.2023.109004.

[240] Y. Tong, D. Shi, Y. Xu, W. Lv, Z. Qin, and X. Tang. Combinatorial Optimization Meets Reinforcement Learning: Effective Taxi Order Dispatching at Large-Scale. *IEEE Transactions on Knowledge and Data Engineering*, 35(10):9812–9823, Oct. 2023. ISSN 1558-2191. doi: 10.1109/TKDE.2021.3127077.

[241] C. Tu, R. Bai, U. Aickelin, Y. Zhang, and H. Du. A deep reinforcement learning hyper-heuristic with feature fusion for online packing problems. *Expert Systems with Applications*, 230:120568, Nov. 2023. ISSN 0957-4174. doi: 10.1016/j.eswa.2023.120568.

[242] A. Turky, N. R. Sabar, S. Dunstall, and A. Song. Hyper-heuristic local search for combinatorial optimisation problems. *Knowledge-Based Systems*, 205:106264, Oct. 2020. ISSN 0950-7051. doi: 10.1016/j.knosys.2020.106264.

[243] W. Vancroonenburg, P. Smet, and G. Vanden Berghe. A two-phase heuristic approach to multi-day surgical case scheduling considering generalized resource constraints. *Operations Research for Health Care*, 7:27–39, Dec. 2015. ISSN 22116923. doi: 10.1016/j.orhc.2015.09.010.

[244] R. Verma, A. Singhal, H. Khadilkar, A. Basumatary, S. Nayak, H. V. Singh, S. Kumar, and R. Sinha. A Generalized Reinforcement Learning Algorithm for Online 3D Bin-Packing. *arXiv:2007.00463 [cs]*, July 2020.

[245] B. Vijayakumar, P. J. Parikh, R. Scott, A. Barnes, and J. Gallimore. A dual bin-packing approach to scheduling surgical cases at a publicly-funded hospital. *European Journal of Operational Research*, 224(3):583–591, Feb. 2013. ISSN 0377-2217. doi: 10.1016/j.ejor.2012.09.010.

[246] M. Vilà and J. Pereira. A branch-and-bound algorithm for assembly line worker assignment and balancing problems. *Computers & Operations Research*, 44:105–114, Apr. 2014. ISSN 0305-0548. doi: 10.1016/j.cor.2013.10.016.

[247] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer Networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[248] Y. Wang, J. Tang, and R. Y. K. Fung. A column-generation-based heuristic algorithm for solving operating theater planning problem under stochastic demand and surgery cancellation risk. *International Journal of Production Economics*, 158:28–36, Dec. 2014. ISSN 0925-5273. doi: 10.1016/j.ijpe.2014.07.015.

[249] Z. Wang and K. Nip. Bin packing under linear constraints. *Journal of Combinatorial Optimization*, 34(4):1198–1209, Nov. 2017. ISSN 1573-2886. doi: 10.1007/s10878-017-0140-2.

[250] G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, Dec. 2007. ISSN 03772217. doi: 10.1016/j.ejor.2005.12.047.

[251] L. Wei, Z. Luo, R. Baldacci, and A. Lim. A New Branch-and-Price-and-Cut Algorithm for One-Dimensional Bin-Packing Problems. *INFORMS Journal on Computing*, 32(2):428–443, Apr. 2020. ISSN 1091-9856. doi: 10.1287/ijoc.2018.0867.

[252] J. M. Weinand, K. Sörensen, P. San Segundo, M. Kleinebrahm, and R. McKenna. Research trends in combinatorial optimization. *International Transactions in Operational Research*, 29(2):667–705, 2022. ISSN 1475-3995. doi: 10.1111/itor.12996.

[253] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in Time Series: A Survey, May 2023.

[254] G. Woeginger. Improved Space for Bounded-Space, On-Line Bin-Packing. *SIAM Journal on Discrete Mathematics*, 6(4):575–581, Nov. 1993. ISSN 0895-4801. doi: 10.1137/0406045.

[255] G. Woeginger. There is no asymptotic PTAS for two-dimensional vector

packing. *Information Processing Letters*, 64(6):293–297, 1997. ISSN 0020-0190. doi: 10.1016/S0020-0190(97)00179-8.

[256] W. Wu, C. Fan, J.-Y. Huang, Z. Liu, and J. Yan. Machine Learning for the Multi-Dimensional Bin Packing Problem: Literature Review and Empirical Evaluation. *arXiv.org*, 2023.

[257] X. Wu, X. Shen, and L. Zhang. Solving the planning and scheduling problem simultaneously in a hospital with a bi-layer discrete particle swarm optimization. *Mathematical biosciences and engineering: MBE*, 16(2):831–861, Jan. 2019. ISSN 1551-0018. doi: 10.3934/mbe.2019039.

[258] Á. S. Xavier, F. Qiu, and S. Ahmed. Learning to Solve Large-Scale Security-Constrained Unit Commitment Problems. *INFORMS Journal on Computing*, page ijoc.2020.0976, Oct. 2020. ISSN 1091-9856, 1526-5528. doi: 10.1287/ijoc.2020.0976.

[259] W. Xiang, J. Yin, and G. Lim. An ant colony optimization approach for solving an operating room surgery scheduling problem. *Computers & Industrial Engineering*, 85:335–345, July 2015. ISSN 0360-8352. doi: 10.1016/j.cie.2015.04.010.

[260] J. Yan, Y. Lu, L. Chen, S. Qin, Y. Fang, Q. Lin, T. Moscibroda, S. Rajmohan, and D. Zhang. Solving the Batch Stochastic Bin Packing Problem in Cloud: A Chance-constrained Optimization Approach. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pages 2169–2179, New York, NY, USA, Aug. 2022. Association for Computing Machinery. ISBN 978-1-4503-9385-0. doi: 10.1145/3534678.3539334.

[261] X. Yang, T. Cui, H. Wang, and Y. Ye. Multiagent Deep Reinforcement Learning for Electric Vehicle Fast Charging Station Pricing Game in

Electricity-Transportation Nexus. *IEEE Transactions on Industrial Informatics*, 20(4):6345–6355, Apr. 2024. ISSN 1941-0050. doi: 10.1109/TII.2023.3345457.

[262] A. C.-C. Yao. New Algorithms for Bin Packing. *Journal of the ACM*, 27 (2):207–227, Apr. 1980. ISSN 0004-5411, 1557-735X. doi: 10.1145/322186.322187.

[263] D. Ye, F. Xie, and G. Zhang. Truthful mechanism design for bin packing with applications on cloud computing. *Journal of Combinatorial Optimization*, 44(4):2224–2245, Nov. 2022. ISSN 1573-2886. doi: 10.1007/s10878-020-00601-4.

[264] G. Yu and G. Zhang. Bin Packing of Selfish Items. In C. Papadimitriou and S. Zhang, editors, *Internet and Network Economics*, pages 446–453, Berlin, Heidelberg, 2008. Springer. ISBN 978-3-540-92185-1. doi: 10.1007/978-3-540-92185-1_50.

[265] H. Yu, W. Liu, J. Lu, Y. Wen, X. Luo, and G. Zhang. Detecting group concept drift from multiple data streams. *Pattern Recognition*, 134:109113, Feb. 2023. ISSN 0031-3203. doi: 10.1016/j.patcog.2022.109113.

[266] L. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, June 1965. ISSN 00199958. doi: 10.1016/S0019-9958(65)90241-X.

[267] C. Zhang and G. Zhang. From packing rules to cost-sharing mechanisms. *Journal of Combinatorial Optimization*, 44(3):1578–1593, Oct. 2022. ISSN 1573-2886. doi: 10.1007/s10878-019-00519-6.

[268] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang. Multitask Genetic Programming-Based Generative Hyperheuristics: A Case Study in Dynamic Scheduling. *IEEE Transactions on Cybernetics*, 52(10):10515–10528, Oct. 2022. ISSN 2168-2275. doi: 10.1109/TCYB.2021.3065340.

[269] G. Zhang and M. Yue. Tight performance bound of AFB k bin packing. *Acta Mathematicae Applicatae Sinica-english Series - ACTA MATH APPL SIN-ENGL SER*, 13:443–446, Oct. 1997. doi: 10.1007/BF02009554.

[270] H. Zhang, R. Bai, T.-Y. Liu, J. Li, B. Lin, and J. Ren. Pattern based learning and optimisation through pricing for bin packing problem, Aug. 2024.

[271] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen. Data-Driven Intelligent Transportation Systems: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1624–1639, 2011. ISSN 1558-0016. doi: 10.1109/TITS.2011.2158001.

[272] J. Zhang, M. Dridi, and A. El Moudni. Column-generation-based heuristic approaches to stochastic surgery scheduling with downstream capacity constraints. *International Journal of Production Economics*, 229:107764, Nov. 2020. ISSN 09255273. doi: 10.1016/j.ijpe.2020.107764.

[273] X. Zhang, M. Fan, D. Wang, P. Zhou, and D. Tao. Top-k Feature Selection Framework Using Robust 0–1 Integer Programming. *IEEE Transactions on Neural Networks and Learning Systems*, 32(7):3005–3019, July 2021. ISSN 2162-2388. doi: 10.1109/TNNLS.2020.3009209.

[274] A. Zhao, T. Li, and L. Lin. A Dynamic Multi-modal deep Reinforcement Learning framework for 3D Bin Packing Problem. *Knowledge-Based Systems*, 299:111990, Sept. 2024. ISSN 0950-7051. doi: 10.1016/j.knosys.2024.111990.

[275] H. Zhao, Q. She, C. Zhu, Y. Yang, and K. Xu. Online 3D Bin Packing with Constrained Deep Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1):741–749, May 2021. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v35i1.16155.

[276] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar. A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, 57(4):99, Mar. 2024. ISSN 1573-7462. doi: 10.1007/s10462-024-10721-6.

[277] H. Zhen, W. Gong, L. Wang, F. Ming, and Z. Liao. Two-Stage Data-Driven Evolutionary Optimization for High-Dimensional Expensive Problems. *IEEE Transactions on Cybernetics*, 53(4):2368–2379, Apr. 2023. ISSN 2168-2275. doi: 10.1109/TCYB.2021.3118783.

[278] S. Zhu, W. Fan, S. Yang, J. Pei, and P. M. Pardalos. Operating room planning and surgical case scheduling: A review of literature. *Journal of Combinatorial Optimization*, 37(3):757–805, Apr. 2019. ISSN 1573-2886. doi: 10.1007/s10878-018-0322-6.

[279] S. Zhu, W. Fan, T. Liu, S. Yang, and P. M. Pardalos. Dynamic three-stage operating room scheduling considering patient waiting time and surgical overtime costs. *Journal of Combinatorial Optimization*, 39(1):185–215, Jan. 2020. ISSN 1382-6905, 1573-2886. doi: 10.1007/s10878-019-00463-5.

[280] W. Zhu, W. Huang, and A. Lim. A prototype column generation strategy for the multiple container loading problem. *European Journal of Operational Research*, 223(1):27–39, Nov. 2012. ISSN 0377-2217. doi: 10.1016/j.ejor.2012.05.039.

[281] M. E. Zonderland, R. J. Boucherie, N. Litvak, and C. L. A. M. Vleggeert-Lankamp. Planning and scheduling of semi-urgent surgeries. *Health Care Management Science*, 13(3):256–267, Sept. 2010. ISSN 1572-9389. doi: 10.1007/s10729-010-9127-6.