Expert Systems With Applications 249 (2024) 123515

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A pattern-based algorithm with fuzzy logic bin selector for online bin packing problem

Bingchen Lin [a,*], Jiawei Li [a], Tianxiang Cui [a], Huan Jin [a], Ruibin Bai [a], Rong Qu [b], Jon Garibaldi [b]

[a] School of Computer Science, University of Nottingham Ningbo China, Ningbo, China
[b] School of Computer Science, University of Nottingham, Nottingham, United Kingdom

ARTICLE INFO

Keywords:
Online bin packing
Planning under uncertainty
Learning for planning and scheduling
Fuzzy logic
Pattern-based planning

ABSTRACT

The online bin packing problem is a well-known optimization challenge that finds application in a wide range of real-world scenarios. In the paper, we propose a novel algorithm called FuzzyPatternPack(FPP), which leverages fuzzy inference and pattern-based predictions of the distribution of item sizes in online bin packing. In comparison to traditional heuristics like BestFit(BF) and FirstFit(FF), as well as the more recent PatternPack(PaP) and ProfilePacking(PrP) algorithm based on online predictions, FPP demonstrates competitive and superior performance in solving various benchmark problems. Particularly, it excels in addressing problems with evolving distributions, making it a promising solution for real-world applications where the item sizes may change over time. This research unveils the promising potential of employing fuzzy logic to effectively address uncertainty in scheduling and planning problems.

1. Introduction

The bin packing problem (BPP) is a classic combinatorial optimization problem heavily studied in computer science, operations research, and logistics. Its real-world applications involve optimizing the utilization of limited resources in various domains such as transportation (An et al., 2021), scheduling (Su et al., 2021), manufacturing (Arbib et al., 2021), storage (Spencer et al., 2019), and networking (Beloglazov & Buyya, 2010).

A one-dimensional BPP aims to use the minimum number of bins of identical size to pack a set of items of different sizes. In its offline version, the sizes of the items are given prior to the packing. Let C denote the capacity of the bins to be used. The problem is to pack all the items into the fewest number of bins with the same capacity. There are N different sizes of items, and each size i has a quantity q_i and a size s_i. Let y_j be a binary variable to indicate whether bin j is used in a solution ($y_j = 1$) or not ($y_j = 0$). Let x_{ij} be the number of times item type i is packed in bin j. The problem can be formulated as follows:

$$\text{minimize} \quad \sum_{j=1}^{U} y_j \tag{1}$$

$$\text{subject to:} \quad \sum_{j=1}^{U} x_{ij} = q_i \qquad \text{for } i = 1, \dots, N \tag{2}$$

$$\sum_{i=1}^{N} s_i x_{ij} \le C y_j \quad \text{for } j = 1, \dots, U \tag{3}$$

where U is the maximal number of possible bins available to use.

In this work, we focus on the online version of this problem, where the items are revealed one by one, and each item must be packed into a bin immediately upon its arrival, without knowing the future sizes of items or even the total number of items.

As simple heuristics, classical online BPP algorithms do not rely on information other than the current items and bins (Coffman et al., 2013). `Any-Fit` algorithms make decisions based on simple rules. `Harmonic`-based algorithms divide items into different groups and adopt a technique to reserve capacity for specific types of items during packing. Conventionally, the worst-case performance would be analyzed for comparison, usually by evaluating the competitive ratio. The competitive ratio of an online algorithm is defined as the worst-case ratio of its cost divided by the optimal cost, over all possible inputs. It is a number not less than 1. A lower competitive ratio means higher worst-case performance.

While effective in certain scenarios, these methods, which are assessed by worst-case performance, may not fully capitalize on valuable insights that could be derived from the regular distributions observed in real-world input item sequences. With the emergence of learning

* Correspondence to: Room 105, Trent Building, 199 Taikang East Road, Ningbo, 315100, China.
E-mail addresses: Bingchen.Lin@nottingham.edu.cn (B. Lin), Jiawei.Li@nottingham.edu.cn (J. Li), Tianxiang.Cui@nottingham.edu.cn (T. Cui), Huan.Jin@nottingham.edu.cn (H. Jin), Ruibin.Bai@nottingham.edu.cn (R. Bai), Rong.Qu@nottingham.ac.uk (R. Qu), Jon.Garibaldi@nottingham.ac.uk (J. Garibaldi).

https://doi.org/10.1016/j.eswa.2024.123515
Received 27 November 2023; Received in revised form 2 February 2024; Accepted 16 February 2024
Available online 17 February 2024

augmented algorithms (LAA), many new online algorithms have been developed. Non-traditionally, these algorithms leverage predictive analytics based on historical data, allowing for more informed decisions, under uncertain environment. Wei and Zhang (2020) reveals that the trade-off between consistency and robustness are intrinsic in the design of LAAs.

There has been many recent research exploring different online algorithms with prediction. Studied online problems include: rent-or-buy (Antoniadis et al., 2021; Bhattacharya & Das, 2022; Drygala et al., 2023), set covering (Bamas et al., 2020; Xu & Zhang, 2022; Zeynali et al., 2021), scheduling (Angelopoulos & Kamali, 2021; Bampis, Dogeas et al., 2022; Lindermayr & Megow, 2022), routing (Bampis, Escoffier et al., 2022; Gouleakis et al., 2023; Kodialam & Lakshman, 2021), network design (Almanza et al., 2021; Jiang et al., 2021), packing (Im et al., 2021), etc. LAA opens up promising possibilities for online BPP algorithms.

Recently, two online BPP algorithms have been developed incorporating the idea of prediction. Both of them also utilize a practically stable `Any-Fit` heuristic as a fallback option for unpredicted items. Angelopoulos et al. (2021, 2022) developed `ProfilePacking(PrP)` catering discrete item sizes. It generates an offline solution according to a prediction and then packs online items into placeholders in the offline solution. Although it performs better than classical heuristics under fixed distribution and foreknown prediction, there is a lack of promising experimental results of the *robustness* under varying distribution and adaptive prediction. The authors admit the need for a more sophisticated mechanism for better adaptability.

Proposed by Lin et al. (2022), `PatternPack(PaP)` adopts a trivial prediction mechanism based on historical items. Designed to tackle continuous input, it discretizes item sizes into types, and forms bin patterns as a combination of different types of items. Based on historical data, it regularly generates plans constructively, to guide online packing adaptively. Its experimentation shows advantages when dealing with some distributions without prior knowledge. However, the discretization and planning mechanism is designed conservatively to avoid bin overflow, which can lead to excessive wasted space in the solution and lowered consistency. Also, the performance of the algorithm largely depends on the chosen parameters. A more flexible mechanism may be needed to reduce waste in patterns and suit more input distributions.

Fuzzy logic is a form of logic that can mimic human decision-making and handle ambiguity. The basic idea is to use fuzzy sets to represent the truth values of statements. A fuzzy set is a set that has a membership function that assigns a degree of membership to each element in the universe of discourse. The degree of membership can range from 0 to 1, where 0 means the element does not belong to the set at all, and 1 means the element fully belongs to the set. For example, a fuzzy set of empty bins might assign a degree of membership of 0.8 to a bin with 5% space occupied, and a degree of membership of 0.4 to a bin with 15% space occupied. Compared to Boolean logic used in all the algorithms above, fuzzy logic has better flexibility and capability to capture and process vital statistical data during the handling of uncertain input.

As a branch of computational intelligence, fuzzy logic has been applied to optimization problems from various aspects. On the one hand, it can leverage expert knowledge and experience. For instance, Cuevas et al. (2020) proposed a directed search strategy, as part of a metaheuristic, based on a Takagi–Sugeno Fuzzy inference system of expert human knowledge. On the other hand, it can better model or handle problems under uncertainty. For example, Hernández et al. (2019) developed a fuzzy logic classifier for an offline 3-D BPP derived from the application in package delivery companies. Drakulić et al. (2021) incorporated fuzzy logic in algorithm design for covering location problems with probabilistic uncertainty involved. Moreover, Pardalos et al. (2013) modeled a list of combinatorial problems under the fuzzy setting.

In this paper, we introduce a new algorithm called `FuzzyPatternPack(FPP)`, which directly extracts patterns from an offline solution and adopts a fuzzy classification system to perform the bin selection process. Experiments are performed to compare its performance with `PaP`, `PrP` and its variants, as well as some classical heuristics. The contributions of this study are twofold:

– Our study introduces an innovative approach to online bin packing problems by drawing inspiration from the impact of prior knowledge on performance. By extracting valuable insights from historical data, the proposed algorithm incorporates patterns to approximate optimal solutions. The integration of fuzzy logic adds a dynamic assessment of bin fitness. This combination of historical data and fuzzy logic represents a novel advancement in tackling online bin packing problems.

– Our study demonstrates the significant contributions of the proposed algorithm through a comprehensive comparison with existing methods, including `PaP`, `PrP` and classical heuristics. The experimental results exhibit the algorithm's superiority, particularly in handling uncertainties inherent in real-world online bin packing scenarios. Notably, the algorithm's capability in adapting to changing distributions, whether due to unpredictably arriving items or evolving sizes, is state-of-the-art, which signifies its potential to address dynamic real-time challenges effectively.

2. Background

2.1. Online bin packing problem

Online bin packing problem has a long history of study. `BestFit(BF)`, `FirstFit(FF)` and $\texttt{Harmonic}_k$ algorithm are three heuristic algorithms for online bin packing problems. As members of the `Any-Fit` family, BF and FF are simple and intuitive algorithms that do not open new bin until an item cannot be packed into any existing bin. They pack each item into the first bin that has enough space for it, either in the order of arrival or in the order of decreasing free space. Based on the idea of bounded-space algorithms (Johnson, 1973) and reservation technique (Yao, 1980), $\texttt{Harmonic}_k$ algorithm was presented. It is a more sophisticated algorithm that classifies the items into k classes based on their sizes and assigns each class to a dedicated set of bins. The algorithm assigns each class, which contains items with sizes in interval $(\frac{1}{k+1}, \frac{1}{k}]$, to a dedicated set of bins, so that each bin can only contain items from one class. This approach effectively guaranteed a better worst-case performance to `Any-Fit`. Numerous modified versions of $\texttt{Harmonic}_k$ have been developed (Lee & Lee, 1985; Ramanan et al., 1989; Richey, 1991; Seiden, 2002). Table 1 lists some key algorithms in a roughly chronological order, with their features, adopted techniques, input requirements, and competitive ratios.

Without prediction, for any online bin packing algorithm, the competitive ratio is proven to be no lower than 1.54278 (Balogh et al., 2021). It is proved that the competitive ratio of FF and BF is 1.7 (Johnson et al., 1974). For the Harmonic family, the most recent algorithm `Advanced-Harmonic` obtained a competitive ratio of 1.58 (Balogh et al., 2021). Note that better worst-case performance does not lead to better average performance. According to Angelopoulos et al. (2022), simple heuristics such as FF and BF achieve near-optimal performance for uniformly distributed input of Coffman et al. (1996) and often surpass, in practice, many online algorithms with better competitive ratios (Kamali & López-Ortiz, 2015).

Many variants of online BPP have also been extensively researched concerning different aspects of real-world applications. Table 2 shows some popular variants. Some variants have also been studied under offline or multi-dimensional settings. We refer readers to Ali et al. (2022), Christensen et al. (2017) and Coffman et al. (2013) for detailed research progress on online BPP-related problems. Researchers have also explored combining features from multiple variants to create a

Table 1
Key existing algorithms for classic 1-D online Bin Packing Problem (BPP).

Algorithm	Feature			Technique		Input		Competitive Ratio	Reference
	Any-Fit	Almost Any-Fit	Bounded-Space	Reservation Technique	Interval Classification	Continuous	Discrete		
Next-Fit(NF)			✓			✓	✓	2	Johnson (1973) and Fisher (1988)
Worst-Fit(WF)	✓					✓	✓	2	Johnson (1973) and Johnson et al. (1974)
Best-Fit(BF)	✓	✓				✓	✓	1.7	
First-Fit(FF)	✓	✓				✓	✓	1.7	
Next-k-Fit(NF_k)			✓			✓	✓	$1.7+3/10(k-1)$	Johnson (1973) and Csirik and Imreh (1989)
Refined-First-Fit(RFF)				✓	✓	✓	✓	1.66	Yao (1980)
$Harmonic_k$			✓	✓	✓	✓	✓	1.69	Lee and Lee (1985)
Advanced-Harmonic			✓	✓	✓	✓	✓	1.58	Balogh et al. (2017)
Profile-Packing(PrP)				✓			✓	See Section 2.2.1	Angelopoulos et al. (2021)
Pattern-Pack(PaP)				✓	✓	✓	✓	See Section 2.2.2	Lin et al. (2022)

more accurate and realistic model tailored for specific application scenarios.

Recently, many data-driven BPP algorithms have been developed with focus on adapting uncertain environment and evaluated by practical experimentation. Tu et al. (2023) developed a deep reinforcement learning (DRL) hyper-heuristic for two online packing problems, where feature fusion is employed to better characterize uncertain environments. Que et al. (2023) proposed a Transformer model, trained by DRL, to solve 3D packing problem. There are also attempts to automatically create algorithms. For example, Asta et al. (2016) proposed a framework to create heuristics via many parameters with a Genetic Algorithm optimizer. Furthermore, new algorithms have been created for a broader range of cutting and packing problems. Sato et al. (2023) developed a separation and compaction algorithm for the two-open dimension nesting problem. Neuenfeldt Júnior et al. (2019) designed a data mining based framework to assess solution quality for the rectangular 2D strip-packing problem.

Packing-related algorithms have been increasingly used to solve more complex and practical production problems. Other than examples mentioned in Introduction, Li et al. (2022) proposed a container loading algorithm with the integration of a heuristic packing algorithm. Bartmeyer et al. (2022) developed an expert system for textile industry, to quickly generate alternative layouts for cutting irregular pieces from raw materials, while avoiding defective areas. Abohamama and Hamouda (2020) designed a hybrid energy–Aware virtual machine placement algorithm for cloud environments, where the formulation of variable sized bin packing problem was adopted. Al-Moalmi et al. (2021) addressed the container placement problem in data centers, considering the placement as vector bin-packing problem. Luo and Rao (2023) developed a heuristic to tackle a special knapsack packing problem arising from aircraft management on the carrier.

2.2. Existing algorithms with prediction

As explained above, most online BPP algorithms were developed with the aim for better worst-case performance. However, two recent algorithms - PrP and PaP - utilize predictions to improve performance in online packing while also preserve an Any−Fit heuristic as a robust fallback option.

2.2.1. ProfilePacking

Proposed in Angelopoulos et al. (2022), PrP algorithm utilizes predictions on the frequency of item sizes. This algorithm employs a profile set as an approximation of the expected input and packs items according to the optimal packing of this profile set. In other words, it treats each item in the optimal solution as a placeholder for incoming items of the same size. Before packing, PrP requires a prediction, which is also called the prefix. The prefix consists of a collection of item sizes and is packed by an offline algorithm. The offline solution is treated as a profile group, which will be filled according to the type of each incoming item.

The algorithm is claimed to have optimal consistency, meaning that it performs best when the predictions are error-free, and near-optimal robustness, meaning that its performance degrades slowly as the prediction error increases. The reported competitive ratio of PrP is $1+(2+5\epsilon)\eta C+\epsilon$, where ϵ is any fixed constant less than 0.2, η represents the prediction error.

Two variants are proposed in Angelopoulos et al. (2021). The Hybrid(λ) variant integrates the classical FF with a parameter $\lambda \in [0,1]$. The value of λ defines the portion of items handled by PaP. For instance, when $\lambda = 0.5$, each of the two algorithms would handle half of the incoming items. This approach is tested and found to have higher robustness compared to PrP. Here we use PrP(λ) to represent both original and Hybrid(λ), as Hybrid(1) is identical to PrP. The other variant is called Adaptive(w), and it is designed from a more practical perspective. Instead of relying on a pre-known distribution "prefix", it replaces it with a dynamically updating prediction based on a "Sliding Window" approach. This involves considering a w of most recently received items to make predictions.

The practical experiments on benchmarks show that PrP, along with its variants, has advantages compared to classical heuristics. However, the placeholder-based mechanism can only handle items that have appeared in the profile. If an incoming item is of a size not appeared in the prediction, called "special items", it would be packed by FF as a fallback option. In their work, all the tests were done under a discrete setting where the input item sizes are integers in the range $(0, 10^2]$. Nevertheless, if sizes of items are in a much wider range, such as $(0, 10^4]$, it would be very likely to have many more "special items" that are forcibly packed by FF, resulting in degrading performance. For time efficiency and simplified implementation, they adopt FirstFit Decreasing(FFD) to generate profile groups, instead of an exact algorithm.

Table 2
Variants of 1-D Online Bin Packing Problem (BPP).

Variants	Difference to classical online BPP			Real-world aspect	Reference
	Objective	Item	Bin		
Dynamic online BPP	/	May be deleted after packed	/	Cloud computing	Gupta et al. (2022)
Relaxed online BPP	/	May be repacked after packed	/	Cloud computing	Gambosi et al. (2000)
Fully dynamic online BPP	/	May be deleted or repacked after packed	/	Cloud computing	Berndt et al. (2020)
Semi-Online online BPP	/	A subset of incoming items visible	/	Packing assembly line	Zhao et al. (2021)
Online open end BPP	/	/	Total size of items per bin may slightly exceed C	Task scheduler	Epstein (2022)
Online BPP with rejection	To minimize bin usage, plus all rejection costs	May be rejected	/	Distributed storage with transfer cost	Dósa and He (2006)
p-Cardinality constrained online BPP	/	/	The number of packed item could not exceed p per bin	Multi-processor scheduling	Balogh et al. (2020)
Class constrained online BPP	/	Additionally tied to a 'color'	Each bin has at most Q colors of items	Video on demand	Xavier and Miyazawa (2008)
Online BPP with conflicts	/	May be conflicting	Cannot be packed into the same bin	Task scheduling with security concern	Epstein and Levin (2008)
Stochastic online BPP	/	Size may follow a stochastic distribution	Size may follow a stochastic distribution	Uncertainty	Gupta and Radovanović (2020)
Online BPP with advice	/	/	Certain level of prior knowledge provided	Decision making with untrusted prediction	Boyar et al. (2016)

2.2.2. PatternPack

PaP, proposed by Lin et al. (2022), also utilizes the idea of prediction and proposes a mechanism that could overcome the "special item" problem. The bin capacity C is divided into N_s sections. Items are categorized based on the section they belong to. An S_j item is any item with size ranging from $(\frac{j-1}{N_s}, \frac{j}{N_s}]$, where $j \in [1, N_s]$. A pattern is made up of one or more types of items. The algorithm would attempt to generate N_p patterns beforehand with a constraint that the maximum size of all items capped at C, starting from $\{S_{N_s}\}$. For example, if $N_s = 4$ and $N_p = 4$, we could form a list of pattern: $\{S_4\}$, $\{S_1, S_3\}$, $\{S_2, S_2\}$, $\{S_1, S_1, S_2\}$.

During the packing process, the input items are recorded regularly into a sample, which is a collection of the at most N_u recently received items. The sample is treated as a prediction and would be cleared when a plan is generated. A plan will be generated when the number of items in the sample reaches N_u. During planning, each pat tern would be assigned a quota to determine how many times the pattern should be used in the next N_u items.

If a pattern consists of t items, the waste after fully packed would be $t \times \frac{C}{N_s}$ in the worst case. This leads to an approximation ratio of $\frac{N_s}{N_s - tC}$. When $N_s = 17$ and $t \leq 3$ for all patterns applied, for example, the approximation ratio for pattern-based packing is $\frac{17}{14} = 1.214286$ given accurate prediction of the item distribution.

During the packing, there are two groups of bins - pattern bins and BestFit bins. For each pattern, N_s queues are constructed for bins waiting for different item types. Opened pattern bins would be waiting in queues until fully packed according to the corresponding patterns. First, for each item, the algorithm attempts to find a pattern bin that is waiting for this item, in the order of patterns. If no pattern bin matches, the algorithm tries to match a pattern by reviewing the pattern list sequentially. A pattern is matched only when it contains this item type. A pattern is available only when it still has a quota. If a pattern is matched and available, a new pattern bin would be opened and used for packing. If no pattern is matched or available, the items would be packed by BF and inserted into the BF bin collection. In this way, items are grouped into limited categories, so unseen items could be packed according to similar items in the prediction.

This algorithm was evaluated with $n = 10^5$ and $C = 10^4$, to imitate continuous input. Higher performance was observed on middle-sized item distributions, compared to BF, under a certain configuration. However, this algorithm has several drawbacks. Firstly, the patterns generated come with wasted space. The more items in a pattern, the more wasted space could occur. Although the priority of patterns has been considered during pattern generation, some pattern bins still have higher wasted space compared to BF. Secondly, the number of automatically generated patterns grows exponentially as N_s increases. This means that using large N_s is computationally unavailable. Additionally, although this algorithm has been designed to adapt to different input distributions, it is difficult to find appropriate parameters - N_s, N_u, and N_p - to obtain reasonable performance across a wide range of distributions.

3. Our proposed method

Algorithm 1 Pattern-based Online Bin Packing Algorithm with Fuzzy Logic Bin Selector

```
1: function Main
2:     initialize the item recorder sample                    ▷ max size N_u
3:     initialize pattern list patterns                       ▷ N_s × n_p of integers
4:     initialize pattern plan plan                           ▷ n_p of integers
5:     initialize bin waiting list l                          ▷ N_s × n_p of multi-sets
6:     initialize avg. item sizes pSizes                      ▷ N_s × n_p of integers
7:     initialize a collection of Best-Fit bins B_b
8:     while there is unpacked item do                        ▷ fetch next item from input
9:         counter ← counter + 1
10:        Push(sample, item)                                 ▷ record item
11:        Pack(plan, patterns, B_b, l, item,...)
12:        if Size(sample) = N_u then
13:            plan, patterns, pSizes ← Update(sample)
14:            push bins in l into B_b                        ▷ pattern bins turn to BF bin
15:            clear sample and l
```

Algorithm 2 Pattern Plan Update

```
1: function UPDATE(sample)
2:     OPT ← an optimal solution of sample
3:     initialize plan
4:     initialize patterns
5:     initialize pSizes
6:     for each bin b in OPT do
7:         initialize curPattern                          ▷ N_s of integers
8:         for each item in b do                          ▷ identify pattern of b
9:             s ← CEIL(item/C/N_s)                       ▷ round-up
10:            curPattern_s ← curPattern_s + 1
11:            curSizes_s ← curSizes_s + item
12:        i ← FINDMATCH(curPattern, patterns)
13:        if i is found then                             ▷ pattern already identified
14:            plan_i ← plan_i + 1
15:        else                                           ▷ new pattern identified
16:            PUSH(patterns, curPattern)
17:            PUSH(plan, 1)
18:            i ← index of the last element in patterns
19:        for j ← 1 to N_s do                            ▷ accumulate item sizes
20:            pSizes_ij ← −pSizes_ij + curSizes_j
21:    for i ← 1 to SIZE(patterns) do
22:        for j ← 1 to N_s do                            ▷ calculate avg. size
23:            pSizes_ij ← −pSizes_ij/plan_i
       return plan, patterns, pSizes
```

FPP improves PaP in pattern generation and bin selection. Algorithm 1 shows the general framework of our proposed approach, which are mostly inherited from PaP. Initially, when an item is received, it is packed into a bin by BF with its size recorded into the sample. After the sample size reaches N_u, a new pattern list and plan would be generated based on the sample. Recall that a pattern is a combination of different types of items; each pattern has a quota to limit usage; a plan is a list of quotas for all extracted patterns. For the next N_u items, they would guide the packing. There are two parameters: sample size N_u and number of sections N_s. N_p is not needed for configuration, as we have a variable number of patterns n_p.

The key data-structures are as follows: *sample* is a vector of maximum length N_u. $sample_i$ represents the size of the ith recorded item. Pattern list *patterns* is a matrix with width of N_s and dynamic row count n_p. $patterns_i$ is the ith pattern; $patterns_{ij}$ is the number of type-j item in the ith pattern. *plan* is an vector of length n_p. $plan_i$ refers to the quota of ith pattern, i.e., how many times the pattern could be applied to bins. Bin waiting list l is a matrix of multi-sets, with width N_s and dynamic row count n_p. l_{ij} is a multiset of bins following ith pattern, waiting for type-j items. The multiplicity of a bin in the multi-set indicates how many items of this type it is waiting for. To calculate patterns and plan, instead of having a computer-generated pattern list with a maximum size of each pattern strictly equal to C, the new algorithm adopts an offline algorithm to work out an optimal solution, from which patterns are extracted. In this way, the pattern list and plan could be seen as a generalized form of the optimal solution, guiding the following packing towards optimal. Algorithm 2 describes the planning process. The bins in optimal solution are grouped based on the type of items they contain. Each group of bins creates a pattern. For each pattern, the frequency of bins would be the quota of the pattern. Also, the average size of each item type of each pattern is logged to aid future decision-making. The extracted patterns have a possibility to cause bin overflow. For example, an offline solution may contain $\{0.48, 0.51\}$ and $\{0.47, 0.52\}$; under $C = 1$ and $N_s = 4$, they belong to pattern $\{S_2, S_3\}$. The pattern requires a space of 1.25 at most, which is greater than C. Trivial approaches, such as placing the pattern bins in queues, may produce invalid results.

To resolve the possibility of bin overflow, a selection mechanism is designed in the packing process (Algorithm 3). Instead of placing opened pattern bins in waiting queues, multi-sets (lists) are used to store bins of a specific pattern, which wait for a specific type of item. Note that duplication is allowed as a pattern may contain several items of the same type. Random access is also not required. For an incoming item, a bin is a potential bin if it is a pattern bin waiting for the item or a BestFit bin with sufficient space. We also virtually create empty pattern bins or BestFit bins. All potential bins with sufficient remaining space would be gathered for selection, based on a fitness value, generated with a Mamdani fuzzy classification system.

Table 3
Input fuzzy variables, where $\overline{X}_{p_b,s_i}$ is a collection of average sizes of items of pattern p_b and section s_i from the optimal solution, I_b is the collection of all items in b after the current item is packed, p_b is the pattern followed by b, s_i is the section index of this item, b is the current bin under fuzzy assessment.

Fuzzy variable	Definition
Waste after packing	$C - \Sigma_{i \in I_b} i$, the wasted space after the item is packed into the bin b.
Diff. after packing	$\Sigma_{i \in I_b} \lvert i - \overline{X}_{p_b,s_i} \rvert$, the difference between b and the optimal bins from which p_b is extracted, after packed.
Remaining quota	Remaining quota of p_b.
Expected waste	The average waste of the extracted optimal bins of this pattern bin.

Algorithm 3 Pack Item

```
1: function PACK(&plan, patterns, &B_b, l, item, ...)
2:     s ← CEIL(item/C/N_s)
3:     B ← FINDALLPOTENTIALBINS(B_b, l, item)
4:     initialize fitness with SIZE(B)
5:     for i ← 1 to SIZE(B) do
6:         fitness_i ← CALCFITNESS(B_i, item, ...)
7:     targetBin ← the bin with the highest fitness value
8:     if targetBin is a BestFit bin then
9:         PACKIN(targetBin, item)
10:        if targetBin is a new bin then
11:            PUSH(B_b, targetBin)
12:    else
13:        PACKIN(targetBin, item)
14:        plan_p ← plan_p − 1
15:        p ← GETPID(targetBin)                          ▷ pattern index
16:        if targetBin is a new bin then
17:            for i ← 1 to N_s do
18:                for j ← 1 to patterns_pi do
19:                    PUSH(l_ij, &targetBin)             ▷ to wait in l
20:        s ← CEIL(item/C/N_s)
21:        ERASE(l_ps, targetBin)
22:        if targetBin has been packed to full then
23:            PUSH(B_b, targetBin)
```

Fitness levels are the defuzzified output of a fuzzy classification process. This process classifies each bin into two categories - 'low' fitness and 'high' fitness. It is based on four input variables (Table 3), of which the membership functions are manually tuned (Fig. 1). The datasets used in tuning is different to the ones used in this paper for evaluation. This process involves three parallel sets of rules, aiming to classify three categories of bins: BestFit bins, existing pattern bins, and new pattern bins.

Table 4 shows the rule sets for the fuzzy classification process. For BestFit bins, the rules are made very similar to the original BF algorithms. The bin with the lowest remaining capacity would gain the highest fitness level. For existing pattern bins, the goal is to maximize similarity to the optimal solution. We only consider "diff. after packing". For new pattern bins, three factors should be considered. The remaining quota is a general pattern-wise guidance. The "diff. after

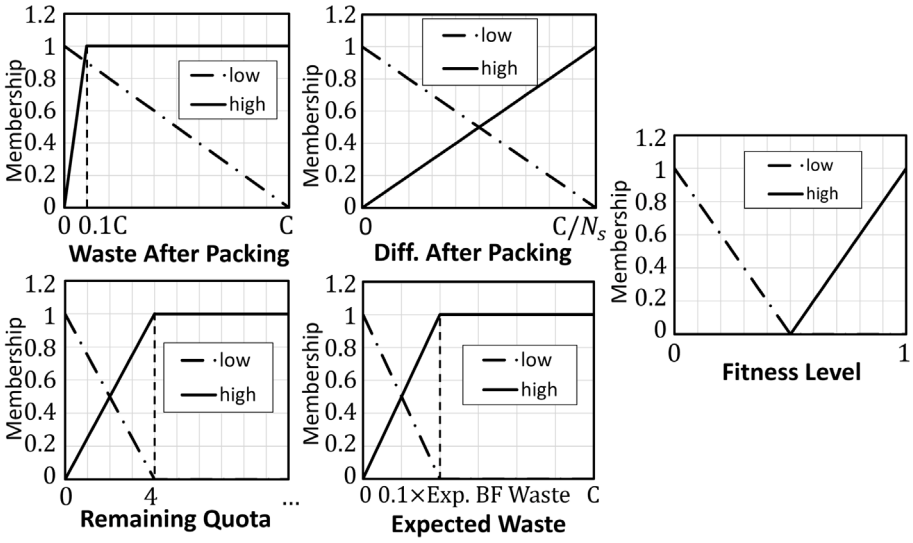

Fig. 1. Membership functions of fuzzy variables.

Table 4
Fuzzy rule matrix (fuzzy associative matrix) for evaluating different type of bins. **(a)** is the rules for BestFit bins, with 1 input. **(b)** is the rules for existing pattern bins, with 1 input. **(c)** is the rules for new pattern bins, with 3 inputs. L and H represent low and high, respectively.

(a)

Waste After Packing	
L	H
H	L

(b)

Diff. After Packing	
L	H
H	L

(c)

		Expected Waste			
		L		H	
		Diff. After Packing		Diff. After Packing	
		L	H	L	H
Remaining Quota	L	L	L	L	L
	H	H	L	L	L

packing" ensures the item could form a similar bin to optimal. The expected waste evaluates the quality of the pattern: if a pattern leads to a higher waste than BF, it should not be used.

The fitness level would be defuzzified by the Centroid method:

$$z^* = \frac{\int \mu_{\widetilde{C}}(z) \cdot z \, \mathrm{d}z}{\int \mu_{\widetilde{C}}(z)\mathrm{d}z}$$

where $\widetilde{C}$ is the fuzzy set, z is a member of the fuzzy set, $\mu_{\widetilde{C}}(z)$ is the membership degree of z.

Due to BPP's NP-Hard nature and time-intensive exact solution search, it is impractical to call an exact solver frequently in exhaustive tests. Similar to PrP, which uses FF for online fallback and FFD for offline solutions in implementation, we adopt `BestFit Decreasing(BFD)` solutions to approximate optimal.

4. Datasets

In this section, we would describe the datasets we used in the problem instance generation. Overall 10 problem instances are generated. The first 4 are locally generated, based on the distribution configuration taken from Lin et al. (2022); the others follows the adoption of Angelopoulos et al. (2021, 2022). For the experiments, we compare our novel algorithm with PaP and PrP separately, with 10 input sequences.

4.1. Instance 1 to 4

Datasets 1 to 4 were used by Lin et al. (2022) to demonstrate the performance of PaP with continuous input item size over fixed distribution. They are regular distributions generated with standard library functions in C++. Since PrP is designed for discrete problem input, we would not test them on these datasets.

Lin et al. (2022) created four problem instances for algorithm evaluation. These problems are based on uniform distribution and triangular distribution. The bin capacity is 10^4, the number of items is set to 10^5. The problem instance is created by C++ standard library functions `mt19937` and `piecewise_linear_distribution`. The configurations are detailed in Table 5.

4.2. Instance 5 to 10

Datasets 5 to 10 were previously adopted by Angelopoulos et al. (2021, 2022) to assess the performance of PrP under discrete input.

4.2.1. Benchmarks

These instances are based on two types of benchmarks. The first type (Dataset 5) is based on the Weibull distribution, and was first studied by Castiñeiras et al. (2012) as a model of several real-world applications of bin packing, e.g., the 2012 ROADEF/EURO Challenge on a data center problem provided by Google and several examination timetabling problems. The Weibull distribution is specified by two parameters: the shape parameter sh and the scale parameter sc (with $sh, sc > 0$). The shape parameter defines the spread of item sizes: lower values indicate greater skew towards smaller items. The scale parameter, informally, has the effect of stretching out the probability density. In our experiments, we chose $sh \in [1.0, 4.0]$. This is because values outside this range result in trivial sequences with items that are generally too small (hence easy to pack) or too large (for which any online algorithm tends to open a new bin). The scale parameter is not critical, since we scale items to the bin capacity; we thus set $sc = 1000$, in accordance with (Castiñeiras et al., 2012).

The second type of benchmarks (Dateset 6-10) is from the BPPLIB Bin Packing Library (Delorme et al., 2018). This is a collection of bin packing benchmarks used in various works on (offline)

Table 5
Specification for instance 1–4.

ID	Distribution type	Parameters($\times 10^3$)			
		Min	Mode	Max	Seed
1	Uniform	3.0	N/A	6.0	1
2	Triangular	3.0	4.5	6.0	0
3	Uniform	2.0	N/A	6.0	1
4	Triangular	2.0	4.0	6.0	0

Table 6
Statistic comparison of bin usage of PaP and FPP across various configurations and test cases. $N_u = 250, n = 10^5$ and $C = 10^4$.

Algorithm	on bin usage	Test case				Average
		(1)	(2)	(3)	(4)	
FPP (36 confs)	Average	47 513	49 502	41 431	42 821	**45 317**
	Std. Dev.	91	53	82	97	**81**
	Surpassing BF	36	36	36	36	**36(100%)**
PaP (1224 confs)	Average	48 760	47 308	45 857	44 406	46 583
	Std. Dev.	696	125	639	477	484
	Surpassing BF	671	695	616	154	534(44%)
	Equivalent to BF	469	479	469	480	474(39%)
	Inferior to BF	84	50	139	590	216(18%)

algorithms for bin packing. We report experimental results for different benchmarks of the BPPLIB Bin Packing Library, in particular the benchmarks "GI" (Gschwind & Irnich, 2016), "Schwerin" (Schwerin & Wäscher, 1997), "Randomly_Generated" (Delorme et al., 2016), "Schoenfield_Hard28" and "Wäscher" (Wäscher & Gau, 1996). Each benchmark consists of multiple problem instances for offline BPP.

4.2.2. Input generation

The problem generator (Kamali, 2021) converts the offline problems above into online input sequences consisting of 10^6 item with bin capacity 100. It is because these datasets could not be directly used without shuffling and normalizing n and C in advance. It scale down each item to the closest integer in $[1, 100]$. Their choice is relevant for applications such as Virtual Machine placement. We generate two classes of input sequences.

Sequences from a fixed distribution. For Weibull benchmarks, the input sequence consists of items generated independently and uniformly at random, for the shape parameter set to $sh = 3.0$. For BPPLIB benchmarks, each item is chosen uniformly and independently at random from the item sizes in one of the benchmark files; this file is also chosen uniformly at random.

Sequences from an evolving distribution. Here, the distribution of the input sequence changes every 50 000 items. Namely, the input sequence is the concatenation of $10^6/50000$ subsequences. For Weibull benchmarks, each subsequence is a Weibull distribution, whose shape parameter is chosen uniformly at random from [1.0,4.0]. For BPPLIB benchmarks, each subsequence is generated by choosing a file uniformly at random, then generating 50 000 items uniformly at random from that specific file.

5. Experimental results

5.1. Comparison to PatternPack

In this experiment, our objective is to uncover enhancements in both performance and insensitivity to parameters, compared to the initial pattern-based online BPP algorithm PaP, under continuous item sizes. The overall performance of the algorithm on numerous parameter settings would be evaluated by comparing the average and standard deviation on the bin usage as well as the comparison to BF and BFD. We will also observe whether there exists a set of parameters that works stably across different test cases.

During the experiment, we employ iterative assessments across all possible settings of N_s and N_p for the PaP algorithm. These evaluations are conducted on the same benchmark configurations as that detailed in Lin et al. (2022), including $n = 10^5$, $N = 250$, and $C = 10^4$. The range for N_s spans from 10 to 45. We have excluded tests with N_s values below 10 due to the considerably reduced performance of PaP in such scenarios. Additionally, tests with N_s values surpassing 45 have been omitted to mitigate the exponential growth of computer-generated patterns. When $N_s > 45$, the potential generation of over 10^5 patterns could lead to impractical computational demands. For the same reason, the scope for N_p encompasses values between $2^{0.5}$ and 2^{17} logarithmically. Note that FPP does not suffer from such computational difficulty as it only extracts patterns appeared in offline solutions.

The results indicate a significant advantage with FPP. Finding an adaptable configuration is easier compared to PaP. Table 6 summarizes the results statistically, across all tested configurations. The performance and stability of FPP largely outperformed PaP, as indicated by lower average and standard deviation of bin usage for each case. All FPP configurations surpassed BF. In contrast, PaP displayed lower and more erratic performance. For instances (1)-(3), it produced moderate results, with most configurations better than BF. Many configurations equaled BF, while some did worse. However, for instance (4), the trend reversed, with a large portion of configurations exhibiting worse performance versus BF.

Fig. 2 depicts three-dimensional response surfaces for the two algorithms across various parameter combinations on the different test cases. It further reveals the superior performance and the insensitivity to parameters of the new algorithm. Concerning PaP, higher values of N_p and N_s generally correlated with enhanced performance over BF; however, exceptions exist, as evidenced in problem (4), where elevated settings led to unsatisfactory results, even underperforming BF. Therefore, PaP is noticeably sensitive to parameters. A single parameter set may not adapt effectively to a wide array of distributions. On the contrary, FPP consistently surpassed BF across all parameter settings. FPP also exhibits diminished sensitivity to fluctuations in N_s, as its results maintained modest variability with varying N_s values. It aligned its results more closely with the offline BFD outcomes, even occasionally surpassing BFD, as observed in the result of problem (4) of Fig. 2.

5.2. Comparison to ProfilePacking

In the experiment, we compare the performance of PrP and FPP both under fixed distribution and evolving distribution, over discrete input. Since the raw algorithm output - bin usage - has a wide range for different test cases, for evaluation, we adopt "performance gap",

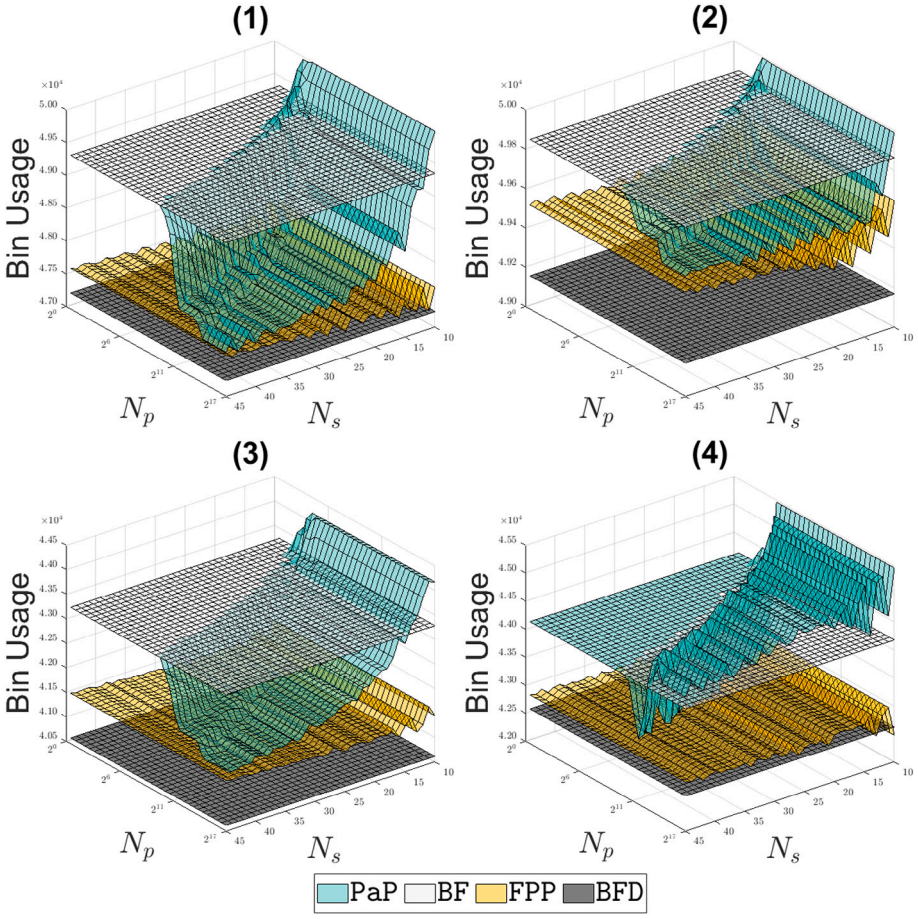

Fig. 2. Bin usage comparison between PaP and FPP, over 4 Test Cases. Results of PaP are plotted as a function of N_p and N_s, results of FPP as a function of N_s, results of BF and BFD as constants. $N_u = 250, n = 10^5$ and $C = 10^4$. The closer the bin usage to BFD, the better.

Table 7
Performance gap comparison for discrete input under fixed distributions among different algorithms. The performance gap is the bin usage more than FFD. PrP is tested with $\lambda = \{0.5, 0.75, 1.0\}$. FPP is tested with $N_s = \{50, 100\}$ and $N_u = 10^4$. A lower gap indicates better performance.

Algorithm	Absolute performance gap to FFD ($\times 10^2$, Bin Usage)					
	Weibull	GI	Schwerin	Rand.	Hard28	Wascher
FFD	0.0	0.0	0.0	0.0	0.0	0.0
FF	129.0	116.1	164.8	144.3	103.4	35.4
BF	126.1	99.2	164.8	124.9	96.4	34.3
PrP(0.5)	77.1	67.5	127.3	82.2	70.3	26.8
PrP(0.75)	50.2	44.0	120.4	51.4	54.3	**22.2**
PrP(1.0)	**23.6**	58.9	**102.6**	57.3	63.8	29.4
FPP(100)	33.6	**36.4**	115.6	**45.0**	**41.3**	25.3
FPP(50)	50.9	**36.4**	120.3	87.6	68.1	31.1

which is the difference of bin usage to an offline baseline algorithm. The gap is assessed in both relative and absolute manner. The closer the performance gap is to 0, the better. We use FFD as the baseline, following Angelopoulos et al. (2022), with problem settings: $n = 10^6$ and $C = 10^2$.

5.2.1. Fixed distributions

These tests are conducted under an ideal setting, where the distribution is stable. The result could illustrate the difference in consistency, i.e. the performance under accurate prediction, between both algorithms.

Two algorithms are tested with various configurations. On the PrP side, the setting is identical to their paper. Both original algorithm and its Hybrid(λ) are tested with $\lambda = \{0.5, 0.75, 1.0\}$. The prediction is generated by randomly selecting items from the input sequence, prior to packing. The most accurate prediction (least error) are taken for comparison. On the FPP side, we tested $N_s = \{50, 100\}$. We use FPP(N_s) to denote FPP with different N_s. We set $N_u = 10^4$, to enable regular pattern extraction from the historical data. Note that, the prediction is solely based on the previously packed items without prior knowledge, i.e. the first 10^4 items are packed by BF.

Table 7 displays the performance gap in the bin usage between the algorithms and FFD. It could be seen that, among all benchmarks, no matter which chosen configuration, PrP and FPP always performed better compared to BF. Generally, the PrP and FPP is similar: both of them obtained the highest results in half of the test cases. FPP(100) achieved the lowest gap in half of the test cases, followed by the prediction-reliant PrP(1.0). The variability in the gap range across test cases is likely attributable to differences in item size distribution for each case.

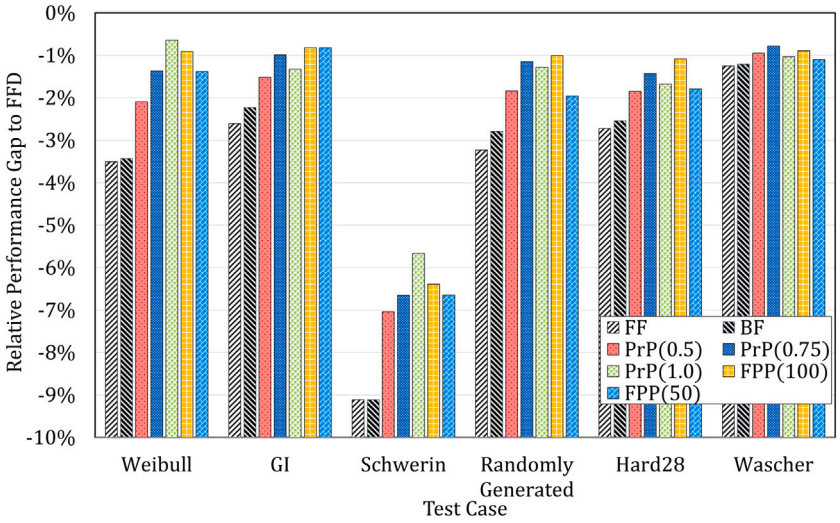

Fig. 3. Performance gap comparison for discrete input under fixed distributions among different algorithms. The gap is the ratio of the difference to FFD in bin usage. PrP is tested with $\lambda = \{0.5, 0.75, 1.0\}$. FPP is tested with $N_s = \{50, 100\}$ and $N_u = 10^4$. The closer the gap is to 0, the better.

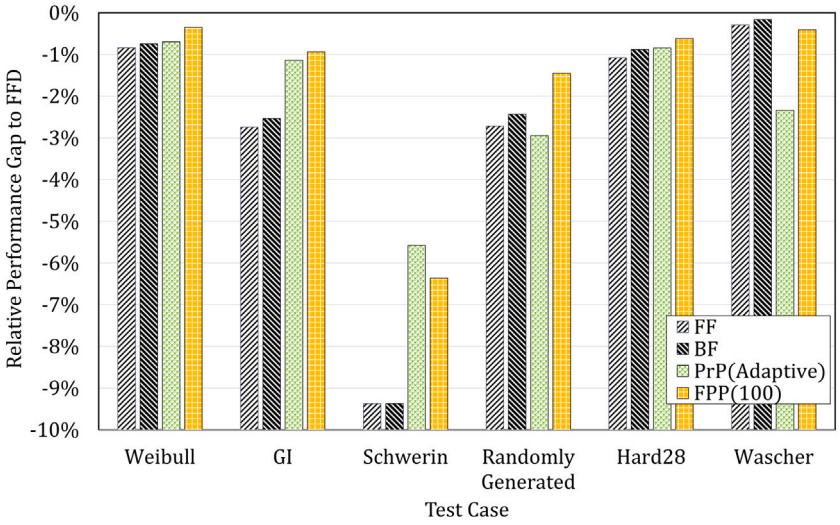

Fig. 4. Performance gap comparison for discrete input under evolving distributions among different algorithms. The performance gap is the ratio of the difference to FFD in bin usage. $w = N_u = 10\,100$. The closer the gap is to 0, the better.

Fig. 3 shows the results as ratio differences in bin usage to FFD on each problem. Among the three PrP configurations, `PrP(0.75)` reached the best performance in a majority of problem instances. However, for Weibull and Schwerin, the most difficult ones for classical heuristics, the original `PrP(1.0)` achieved the highest performance. One possible reason is that, while gaining robustness from classical heuristics, the performance is partly tied to FF, and sacrifices some consistency.

As FPP integrates the classical algorithm BF, a connection to traditional heuristics is also evident, albeit to a lesser extent. Notably, in cases like the Schwerin benchmarks, FPP fell short in competing against `PrP(1.0)`, which heavily relies on predictions. However, it exhibited superior performance compared to `PrP(0.5)` and `PrP(0.75)`, which rely more on FF in packing. This can be attributed to its more advanced bin selection process. Between 2 FPP configurations, `FPP(100)` produced better or equal performance in all instances, compared to `FPP(50)`. It may be due to the discrete problem setting, where item sizes are small integers. For a low C value, such as 10^2, FPP could achieve good performance without loosening the offline solution.

5.2.2. Evolving distributions

We also test algorithms' adaptability and robustness using dynamic distributions that evolve regularly, approximating real-world conditions. The non-stationary distribution enabled comparison of each method's capability to adjust predictions and maintain performance when the historical data is inaccurate to represent future conditions.

We first compare the performance of both algorithms over different instances. As w and N_u both define the size of prediction, we use them interchangeably in this section. They are both set to 10 100, where both algorithms have reasonable performance. Then, we compare the performance of both algorithms as functions of N_u, under different test cases. 100 different values of N_u are tested, equidistant in $[100, 100100]$. Here we report the results of evolving Hard28 distribution, which shows the most typical trends.

Fig. 4 shows the result of the experiment under evolving distributions, as a better imitation of real-world problem input. It could

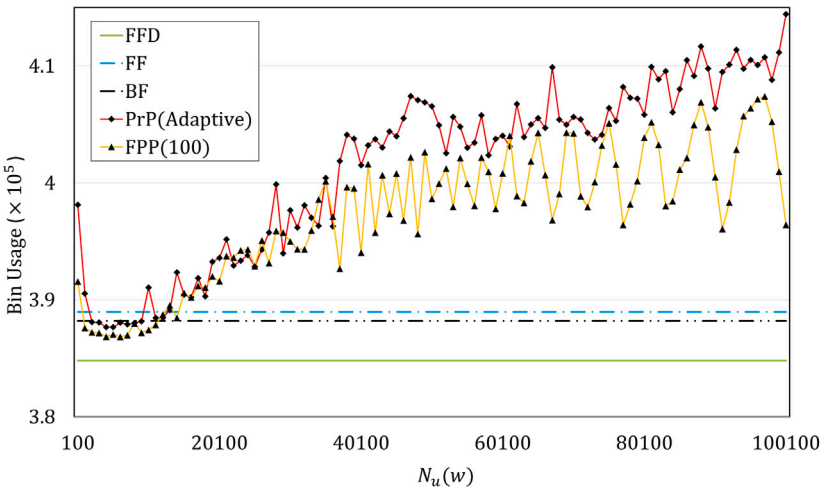

Fig. 5. Bin usage of `PrP(Adaptive)`, `FPP(100)`, FF, BF and FFD under evolving distribution Hard28, as a Function of N_u, i.e. w.

be seen that, for the two algorithms with prediction, the performance advantage over BF was significantly reduced, or even eliminated. For the Wascher instance, BF even had better performance than both of the two algorithms. As BF is a practically robust algorithm, a possible explanation for the above changes is that the prediction accuracy affects the robustness of the two algorithms. When the prediction is not sufficiently accurate, the performance of both algorithms degrades, in some cases, very severely. Additionally, FPP obtained better results, compared to PrP, in a majority of instances (5 out of 6). As both algorithms packs according to the optimal solution of historical data, the pattern extraction and fuzzy bin selection mechanism have better performance in tackling degraded prediction accuracy, compared to trivially converting items in the offline solution into placeholders.

Fig. 5 shows the comparison of two algorithms as the N_u increases. We show the result of instance Hard28 as it includes a majority of common characteristics of all tests. As could be seen, when N_u is small, the performance of both algorithms was lower than classical heuristics. This is because the algorithm does not have sufficient data to construct a correct distribution in the profile or sample. The performance of both prediction-based algorithms reached its highest when N_u is approximately 8000. Since the input distribution was updated every 50 000 items, this could be seen as a balance point between the correctness and promptness of the prediction. As N_u gets larger than the balance point, both algorithms suffer performance degradation. There could be two reasons to explain. First, the prediction is updated slower than needed and thus failed to gain sufficient advantage to outperform classical advantage timely. Second, the samples from the previous distribution poison the correctness of the prediction. Notably, although FPP has more fluctuation with large N_u, it showed an advantage in the vast majority of values of N_u. In most other instances, this trend could also be found. It indicates this algorithm's potential to better handle online BPP uncertainty with varying input distribution.

6. Conclusions and discussions

In real-world online bin packing problems, information on item sizes beforehand can lead to significant performance improvements. Motivated by this, we extract valuable information from historical data and incorporate it into our packing algorithms. In this paper, we introduce a novel approach that leverages patterns to approximate optimal solutions and utilizes fuzzy logic to assess the fitness of potential bins.

We have conducted a comprehensive comparison of our proposed algorithm with PaP, PrP and variants of PrP, as well as classical heuristics, using various benchmark datasets featuring both fixed and evolving distributions. Through extensive experimentation, the results demonstrate the superior performance and robustness of our algorithm over PaP in all cases, as well as PrP in the majority of cases. Notably, our algorithm exhibits exceptional capabilities in handling uncertainty and variability inherent in online bin packing problems.

In scenarios with fixed distributions, our algorithm consistently outperforms PaP and the classical heuristics, showcasing its ability to find better solutions efficiently. Moreover, the utilization of historical data and the incorporation of fuzzy logic enables our algorithm to adapt dynamically to changing distributions in real-time scenarios, where items may arrive unpredictably, or their sizes may evolve over time, which is state-of-the-art. These results affirm the promising potential of our algorithm in solving online bin packing problems, making it a highly practical and effective solution for a wide range of real-world applications.

Future work could be done in several directions. Firstly, a comprehensive analysis of the algorithm's competitivity could be explored to establish a more solid theoretical ground. Secondly, the fuzzy classification system can be refined through varied membership functions and inference rules, automated methods for component optimization, and simplifying rules into equations. Further investigation could be pursued to broaden fuzzification, encompassing a wider array of uncertain information. Thirdly, the algorithm could be extended to handle more complex online problems.

CRediT authorship contribution statement

Bingchen Lin: Conceptualization, Software, Methodology, Investigation, Writing – original draft, Visualization. **Jiawei Li:** Conceptualization, Methodology, Formal analysis, Resources, Supervision, Writing – review & editing, Funding acquisition. **Tianxiang Cui:** Writing – review & editing. **Huan Jin:** Supervision. **Ruibin Bai:** Resources. **Rong Qu:** Writing – review & editing. **Jon Garibaldi:** Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Ruibin Bai reports financial support was provided by National Natural Science Foundation of China. Ruibin Bai reports financial support was provided by Ningbo Science and Technology Bureau. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This research is supported by a NSFC project of China (72071116) and a Ningbo Science and Technology Bureau 2025 Major Projects Fund, China (2021Z089).

Appendix. Nomenclature

List of Symbols

Parameters

C	Bin capacity
N	The number of different sizes of items
U	The maximal number of possible bins available to use
sh	Shape parameter of Weibull distribution
n	The number of items
λ	The portion of items handled by PrP
k	The number of partitions in Harmonic
w	The sliding windows width of adaptive variant of PrP
N_s	The number of sections in PaP and FPP
N_p	The number of patterns in PaP
N_u	The sample size in PaP and FPP
sc	Scale parameter of Weibull distribution

Variables

t	The number of items in a pattern
ϵ	Any fixed constant less than 0.2
η	The prediction error
q_i	Quantity of size i
s_i	Size of size i
x_{ij}	The number of times item type i is packed in bin j
y_j	Whether bin j is used in a solution
j	A section index
S_j	The jth section, $(\frac{j-1}{N_s}, \frac{j}{N_s}]$
n_p	The number of patterns in FPP
b	The current bin under fuzzy assessment
$i \in I_b$	Items in b after the current item is packed
p_b	The pattern followed by b
s_i	The section that item i belongs
$\overline{X}_{p_b,s_i}$	Average size of items of p_b and s_i of optimal solution
$\widetilde{C}$	The fuzzy set
z	A member of $\widetilde{C}$
$\widetilde{C}(z)$	The membership degree of z in $\widetilde{C}$
z^*	The defuzzified value of z

References

Abohamama, A. S., & Hamouda, E. (2020). A hybrid energy–Aware virtual machine placement algorithm for cloud environments. *Expert Systems with Applications*, *150*, Article 113306. http://dx.doi.org/10.1016/j.eswa.2020.113306, URL: https://www.sciencedirect.com/science/article/pii/S0957417420301317.

Al-Moalmi, A., Luo, J., Salah, A., Li, K., & Yin, L. (2021). A whale optimization system for energy-efficient container placement in data centers. *Expert Systems with Applications*, *164*, Article 113719. http://dx.doi.org/10.1016/j.eswa.2020.113719, URL: https://www.sciencedirect.com/science/article/pii/S0957417420305431.

Ali, S., Ramos, A. G., Carravilla, M. A., & Oliveira, J. F. (2022). On-line three-dimensional packing problems: A review of off-line and on-line solution approaches. *Computers & Industrial Engineering*, *168*, Article 108122. http://dx.doi.org/10.1016/j.cie.2022.108122, URL: https://www.sciencedirect.com/science/article/pii/S0360835222001929.

Almanza, M., Chierichetti, F., Lattanzi, S., Panconesi, A., & Re, G. (2021). Online facility location with multiple advice. *Vol. 34*, In *Advances in neural information processing systems* (pp. 4661–4673). Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2021/hash/250473494b245120a7eaf8b2e6b1f17c-Abstract.html.

An, N. H., Phát, N. T., & Tú, L. T. N. (2021). A novel threshold based approach of detecting oil spills on sea in synthetic aperture radar images. *TNU Journal of Science and Technology*, *226*(06), 10–17. http://dx.doi.org/10.34238/tnu-jst.3839, URL: http://jst.tnu.edu.vn/jst/article/view/3839.

Angelopoulos, S., & Kamali, S. (2021). Contract Scheduling With Predictions. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(13), 11726–11733. http://dx.doi.org/10.1609/aaai.v35i13.17394, URL: https://ojs.aaai.org/index.php/AAAI/article/view/17394. Section: AAAI Technical Track on Planning, Routing, and Scheduling.

Angelopoulos, S., Kamali, S., & Shadkami, K. (2021). Online bin packing with predictions. URL: http://arxiv.org/abs/2102.03311. arXiv:2102.03311 [cs].

Angelopoulos, S., Kamali, S., & Shadkami, K. (2022). Online bin packing with predictions. In *Proceedings of the thirty-first international joint conference on artificial intelligence* (pp. 4574–4580). Vienna, Austria: International Joint Conferences on Artificial Intelligence Organization, http://dx.doi.org/10.24963/ijcai.2022/635, URL: https://www.ijcai.org/proceedings/2022/635.

Antoniadis, A., Coester, C., Elias, M., Polak, A., & Simon, B. (2021). Learning-augmented dynamic power management with multiple states via new ski rental bounds. *Vol. 34*, In *Advances in neural information processing systems* (pp. 16714–16726). Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2021/hash/8b8388180314a337c9aa3c5aa8e2f37a-Abstract.html.

Arbib, C., Marinelli, F., & Pizzuti, A. (2021). Number of bins and maximum lateness minimization in two-dimensional bin packing. *European Journal of Operational Research*, *291*(1), 101–113. http://dx.doi.org/10.1016/j.ejor.2020.09.023, URL: https://www.sciencedirect.com/science/article/pii/S0377221720308274.

Asta, S., Özcan, E., & Parkes, A. J. (2016). CHAMP: Creating heuristics via many parameters for online bin packing. *Expert Systems with Applications*, *63*, 208–221. http://dx.doi.org/10.1016/j.eswa.2016.07.005, URL: https://www.sciencedirect.com/science/article/pii/S0957417416303499.

Balogh, J., Békési, J., Dósa, G., Epstein, L., & Levin, A. (2017). A new and improved algorithm for online bin packing. URL: http://arxiv.org/abs/1707.01728. arXiv:1707.01728 [cs, math].

Balogh, J., Békési, J., Dósa, G., Epstein, L., & Levin, A. (2020). Online bin packing with cardinality constraints resolved. *Journal of Computer and System Sciences*, *112*, 34–49. http://dx.doi.org/10.1016/j.jcss.2020.03.002, URL: https://www.sciencedirect.com/science/article/pii/S0022000018300722.

Balogh, J., Békési, J., Dósa, G., Epstein, L., & Levin, A. (2021). A new lower bound for classic online bin packing. *Algorithmica*, *83*(7), 2047–2062. http://dx.doi.org/10.1007/s00453-021-00818-7.

Bamas, E., Maggiori, A., & Svensson, O. (2020). The primal-dual method for learning augmented algorithms. *Vol. 33*, In *Advances in neural information processing systems* (pp. 20083–20094). Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2020/hash/e834cb114d33f729dbc9c7fb0c6bb607-Abstract.html.

Bampis, E., Dogeas, K., Kononov, A., Lucarelli, G., & Pascual, F. (2022). *Vol. 5*, *Scheduling with untrusted predictions* (pp. 4581–4587). http://dx.doi.org/10.24963/ijcai.2022/636, URL: https://www.ijcai.org/proceedings/2022/636. ISSN: 1045-0823.

Bampis, E., Escoffier, B., & Xefteris, M. (2022). Canadian Traveller Problem with Predictions. In P. Chalermsook, & B. Laekhanukit (Eds.), *Lecture notes in computer science*, *Approximation and online algorithms* (pp. 116–133). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-031-18367-6_6.

Bartmeyer, P. M., Oliveira, L. T., Leão, A. A. S., & Toledo, F. M. B. (2022). An expert system to react to defective areas in nesting problems. *Expert Systems with Applications*, *209*, Article 118207. http://dx.doi.org/10.1016/j.eswa.2022.118207, URL: https://www.sciencedirect.com/science/article/pii/S095741742201363X.

Beloglazov, A., & Buyya, R. (2010). Energy efficient resource management in virtualized cloud data centers. In *2010 10th IEEE/ACM international conference on cluster, cloud and grid computing* (pp. 826–831). http://dx.doi.org/10.1109/CCGRID.2010.46.

Berndt, S., Jansen, K., & Klein, K.-M. (2020). Fully dynamic bin packing revisited. *Mathematical Programming*, *179*(1), 109–155. http://dx.doi.org/10.1007/s10107-018-1325-x.

Bhattacharya, A., & Das, R. (2022). Machine learning advised ski rental problem with a discount. In P. Mutzel, M. S. Rahman, & Slamin (Eds.), *Lecture notes in computer science*, *WALCOM: algorithms and computation* (pp. 213–224). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-96731-4_18.

Boyar, J., Favrholdt, L. M., Kudahl, C., Larsen, K. S., & Mikkelsen, J. W. (2016). Online algorithms with advice: A survey. *ACM SIGACT News*, *47*(3), 93–129. http://dx.doi.org/10.1145/2993749.2993766, URL: https://dl.acm.org/doi/10.1145/2993749.2993766.

Castiñeiras, I., De Cauwer, M., & O'Sullivan, B. (2012). Weibull-based benchmarks for bin packing. In M. Milano (Ed.), *Lecture notes in computer science*, *Principles and practice of constraint programming* (pp. 207–222). Berlin, Heidelberg: Springer, http://dx.doi.org/10.1007/978-3-642-33558-7_17.

Christensen, H. I., Khan, A., Pokutta, S., & Tetali, P. (2017). Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, *24*, 63–79. http://dx.doi.org/10.1016/j.cosrev.2016.12.001, URL: https://linkinghub.elsevier.com/retrieve/pii/S1574013716301356.

Coffman, E. G., Csirik, J., Galambos, G., Martello, S., & Vigo, D. (2013). Bin packing approximation algorithms: Survey and classification. In P. M. Pardalos, D.-Z. Du, & R. L. Graham (Eds.), *Handbook of combinatorial optimization* (pp. 455–531). New York, NY: Springer New York, http://dx.doi.org/10.1007/978-1-4419-7997-1_35, URL: https://link.springer.com/10.1007/978-1-4419-7997-1_35.

Coffman, E. G., Garey, M. R., & Johnson, D. S. (1996). Approximation algorithms for bin packing: A survey. In *Approximation algorithms for NP-hard problems* (pp. 46–93). USA: PWS Publishing Co..

Csirik, J., & Imreh, B. (1989). On the worst-case performance of the NkF bin-packing heuristic. *Acta Cybernetica*, *9*(2), 89–105, URL: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3358. Number: 2 Publisher: University of Szeged.

Cuevas, E., Gálvez, J., & Avalos, O. (2020). Fuzzy logic based optimization algorithm. In *Studies in computational intelligence*: *vol. 854*, *Recent metaheuristics algorithms for parameter identification* (pp. 135–181). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-28917-1_6, URL: http://link.springer.com/10.1007/978-3-030-28917-1_6.

Delorme, M., Iori, M., & Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, *255*(1), 1–20. http://dx.doi.org/10.1016/j.ejor.2016.04.030, URL: https://www.sciencedirect.com/science/article/pii/S0377221716302491.

Delorme, M., Iori, M., & Martello, S. (2018). BPPLIB: a library for bin packing and cutting stock problems. *Optimization Letters*, *12*(2), 235–250. http://dx.doi.org/10.1007/s11590-017-1192-z, URL: http://link.springer.com/10.1007/s11590-017-1192-z.

Dósa, G., & He, Y. (2006). Bin packing problems with rejection penalties and their dual problems. *Information and Computation*, *204*(5), 795–815. http://dx.doi.org/10.1016/j.ic.2006.02.003, URL: https://www.sciencedirect.com/science/article/pii/S0890540106000228.

Drakulić, D., Takači, A., & Marić, M. (2021). The use of fuzzy logic in various combinatorial optimization problems. In E. Pap (Ed.), *Studies in computational intelligence*: *vol. 973*, *Artificial intelligence: theory and applications* (pp. 137–153). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-72711-6_8, URL: https://link.springer.com/10.1007/978-3-030-72711-6_8.

Drygala, M., Nagarajan, S. G., & Svensson, O. (2023). Online algorithms with costly predictions. In F. Ruiz, J. Dy, & J.-W. van de Meent (Eds.), *Proceedings of machine learning research*: *vol. 206*, *Proceedings of the 26th international conference on artificial intelligence and statistics* (pp. 8078–8101). PMLR, URL: https://proceedings.mlr.press/v206/drygala23a.html.

Epstein, L. (2022). Open-end bin packing: New and old analysis approaches. *Discrete Applied Mathematics*, *321*, 220–239. http://dx.doi.org/10.1016/j.dam.2022.07.003, URL: https://www.sciencedirect.com/science/article/pii/S0166218X22002372.

Epstein, L., & Levin, A. (2008). On bin packing with conflicts. *SIAM Journal on Optimization*, *19*(3), 1270–1298. http://dx.doi.org/10.1137/060666329, URL: https://epubs.siam.org/doi/abs/10.1137/060666329. Publisher: Society for Industrial and Applied Mathematics.

Fisher, D. C. (1988). Next-fit packs a list and its reverse into the same number of bins. *Operations Research Letters*, *7*(6), 291–293. http://dx.doi.org/10.1016/0167-6377(88)90060-0, URL: https://www.sciencedirect.com/science/article/pii/0167637788900600.

Gambosi, G., Postiglione, A., & Talamo, M. (2000). Algorithms for the relaxed online bin-packing model. *SIAM Journal on Computing*, *30*(5), 1532–1551. http://dx.doi.org/10.1137/S0097539799180408, URL: https://epubs.siam.org/doi/10.1137/S0097539799180408. Publisher: Society for Industrial and Applied Mathematics.

Gouleakis, T., Lakis, K., & Shahkarami, G. (2023). Learning-augmented algorithms for online TSP on the line. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(10), 11989–11996. http://dx.doi.org/10.1609/aaai.v37i10.26414, URL: https://ojs.aaai.org/index.php/AAAI/article/view/26414. Section: AAAI Technical Track on Planning, Routing, and Scheduling.

Gschwind, T., & Irnich, S. (2016). Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, *28*(1), 175–194. http://dx.doi.org/10.1287/ijoc.2015.0670.

Gupta, N., Gupta, K., Gupta, D., Juneja, S., Turabieh, H., Dhiman, G., Kautish, S., & Viriyasitavat, W. (2022). Enhanced virtualization-based dynamic bin-packing optimized energy management solution for heterogeneous clouds. *Mathematical Problems in Engineering*, *2022*, Article e8734198. http://dx.doi.org/10.1155/2022/8734198, URL: https://www.hindawi.com/journals/mpe/2022/8734198/. Publisher: Hindawi.

Gupta, V., & Radovanović, A. (2020). Interior-point-based online stochastic bin packing. *Operations Research*, *68*(5), 1474–1492. http://dx.doi.org/10.1287/opre.2019.1914, URL: https://pubsonline.informs.org/doi/10.1287/opre.2019.1914. Publisher: INFORMS.

Hernández, P. H., Castillo-García, N., Larkins, E. R., Ruiz, J. G. G., Díaz, S. V. M., & Resendiz, E. S. (2019). A fuzzy logic classifier for the three dimensional bin packing problem deriving from package delivery companies application. In *Handbook of research on metaheuristics for order picking optimization in warehouses to smart cities* (pp. 433–442). IGI Global, http://dx.doi.org/10.4018/978-1-5225-8131-4.ch025.

Im, S., Kumar, R., Montazer Qaem, M., & Purohit, M. (2021). Online knapsack with frequency predictions. *Vol. 34*, In *Advances in neural information processing systems* (pp. 2733–2743). Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2021/hash/161c5c5ad51fcc884157890511b3c8b0-Abstract.html.

Jiang, S. H.-C., Liu, E., Lyu, Y., Tang, Z. G., & Zhang, Y. (2021). Online facility location with predictions. URL: https://openreview.net/forum?id=DSQHjibtgKR.

Johnson, D. S. (1973). *Near-optimal bin packing algorithms* (Ph.D. thesis), Massachusetts Institute of Technology.

Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., & Graham, R. L. (1974). Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, *3*(4), 299–325. http://dx.doi.org/10.1137/0203025, URL: http://epubs.siam.org/doi/10.1137/0203025.

Kamali, S. (2021). GitHub - shahink84/BinPackingPredictions: Experiments on online bin packing algorithms with erroneous predictions. URL: https://github.com/shahink84/BinPackingPredictions.

Kamali, S., & López-Ortiz, A. (2015). All-around near-optimal solutions for the online bin packing problem. In K. Elbassioni, & K. Makino (Eds.), *Lecture notes in computer science*, *Algorithms and computation* (pp. 727–739). Berlin, Heidelberg: Springer, http://dx.doi.org/10.1007/978-3-662-48971-0_61.

Kodialam, M., & Lakshman, T. V. (2021). Prediction augmented segment routing. In *2021 IEEE 22nd international conference on high performance switching and routing* (pp. 1–6). http://dx.doi.org/10.1109/HPSR52026.2021.9481858.

Lee, C. C., & Lee, D. T. (1985). A simple on-line bin-packing algorithm. *Journal of the ACM*, *32*(3), 562–572. http://dx.doi.org/10.1145/3828.3833, URL: https://dl.acm.org/doi/10.1145/3828.3833.

Li, Y., Chen, M., & Huo, J. (2022). A hybrid adaptive large neighborhood search algorithm for the large-scale heterogeneous container loading problem. *Expert Systems with Applications*, *189*, Article 115909. http://dx.doi.org/10.1016/j.eswa.2021.115909, URL: https://www.sciencedirect.com/science/article/pii/S0957417421012641.

Lin, B., Li, J., Bai, R., Qu, R., Cui, T., & Jin, H. (2022). Identify patterns in online bin packing problem: An adaptive pattern-based algorithm. *Symmetry*, *14*(7), 1301. http://dx.doi.org/10.3390/sym14071301, URL: https://www.mdpi.com/2073-8994/14/7/1301.

Lindermayr, A., & Megow, N. (2022). Permutation predictions for non-clairvoyant scheduling. In *Proceedings of the 34th ACM symposium on parallelism in algorithms and architectures* (pp. 357–368). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3490148.3538579, event-place: Philadelphia, PA, USA.

Luo, Q., & Rao, Y. (2023). Heuristic algorithms for the special knapsack packing problem with defects arising in aircraft arrangement. *Expert Systems with Applications*, *215*, Article 119392. http://dx.doi.org/10.1016/j.eswa.2022.119392, URL: https://www.sciencedirect.com/science/article/pii/S0957417422024101.

Neuenfeldt Júnior, A., Silva, E., Gomes, A. M., Soares, C., & Oliveira, J. F. (2019). Data mining based framework to assess solution quality for the rectangular 2D strip-packing problem. *Expert Systems with Applications*, *118*, 365–380. http://dx.doi.org/10.1016/j.eswa.2018.10.006, URL: https://www.sciencedirect.com/science/article/pii/S0957417418306481.

Pardalos, P. M., Aydogan, E. K., Gurbuz, F., Demirtas, O., & Bakirli, B. B. (2013). Fuzzy combinatorial optimization problems. In P. M. Pardalos, D.-Z. Du, & R. L. Graham (Eds.), *Handbook of combinatorial optimization* (pp. 1357–1413). New York, NY: Springer New York, http://dx.doi.org/10.1007/978-1-4419-7997-1_68, URL: https://link.springer.com/10.1007/978-1-4419-7997-1_68.

Que, Q., Yang, F., & Zhang, D. (2023). Solving 3D packing problem using Transformer network and reinforcement learning. *Expert Systems with Applications*, *214*, Article 119153. http://dx.doi.org/10.1016/j.eswa.2022.119153, URL: https://www.sciencedirect.com/science/article/pii/S0957417422021716.

Ramanan, P., Brown, D. J., Lee, C. C., & Lee, D. T. (1989). On-line bin packing in linear time. *Journal of Algorithms*, *10*(3), 305–326. http://dx.doi.org/10.1016/0196-6774(89)90031-X, URL: https://www.sciencedirect.com/science/article/pii/019667748990031X.

Richey, M. B. (1991). Improved bounds for harmonic-based bin packing algorithms. *Discrete Applied Mathematics*, *34*(1), 203–227. http://dx.doi.org/10.1016/0166-218X(91)90087-D, URL: https://www.sciencedirect.com/science/article/pii/0166218X9190087D.

Sato, A. K., Mundim, L. R., Martins, T. C., & Tsuzuki, M. S. G. (2023). A separation and compaction algorithm for the two-open dimension nesting problem using penetration-fit raster and obstruction map. *Expert Systems with Applications*, *220*, Article 119716. http://dx.doi.org/10.1016/j.eswa.2023.119716, URL: https://www.sciencedirect.com/science/article/pii/S0957417423002178.

Schwerin, P., & Wäscher, G. (1997). The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP. *International Transactions in Operational Research*, *4*(5), 377–389. http://dx.doi.org/10.1016/S0969-6016(97)00025-7, URL: https://www.sciencedirect.com/science/article/pii/S0969601697000257.

Seiden, S. S. (2002). On the online bin packing problem. *Journal of the ACM*, *49*(5), 640–671. http://dx.doi.org/10.1145/585265.585269, URL: https://dl.acm.org/doi/10.1145/585265.585269.

Spencer, K. Y., Tsvetkov, P. V., & Jarrell, J. J. (2019). A greedy memetic algorithm for a multiobjective dynamic bin packing problem for storing cooling objects. *Journal of Heuristics*, *25*(1), 1–45. http://dx.doi.org/10.1007/s10732-018-9382-0.

Su, B., Xie, N., & Yang, Y. (2021). Hybrid genetic algorithm based on bin packing strategy for the unrelated parallel workgroup scheduling problem. *Journal of Intelligent Manufacturing*, *32*(4), 957–969. http://dx.doi.org/10.1007/s10845-020-01597-8.

Tu, C., Bai, R., Aickelin, U., Zhang, Y., & Du, H. (2023). A deep reinforcement learning hyper-heuristic with feature fusion for online packing problems. *Expert Systems with Applications*, *230*, Article 120568. http://dx.doi.org/10.1016/j.eswa.2023.120568, URL: https://www.sciencedirect.com/science/article/pii/S0957417423010709.

Wäscher, G., & Gau, T. (1996). Heuristics for the integer one-dimensional cutting stock problem: A computational study. *Operations-Research-Spektrum*, *18*(3), 131–144. http://dx.doi.org/10.1007/BF01539705.

Wei, A., & Zhang, F. (2020). Optimal robustness-consistency trade-offs for learning-augmented online algorithms. *Vol. 33*, In *Advances in neural information processing systems* (pp. 8042–8053). Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2020/hash/5bd844f11fa520d54fa5edec06ea2507-Abstract.html.

Xavier, E. C., & Miyazawa, F. K. (2008). The class constrained bin packing problem with applications to video-on-demand. *Theoretical Computer Science*, *393*(1), 240–259. http://dx.doi.org/10.1016/j.tcs.2008.01.001, URL: https://www.sciencedirect.com/science/article/pii/S0304397508000273.

Xu, C., & Zhang, G. (2022). Learning-augmented algorithms for online subset sum. *Journal of Global Optimization*, http://dx.doi.org/10.1007/s10898-022-01156-w.

Yao, A. C.-C. (1980). New algorithms for bin packing. *Journal of the ACM*, *27*(2), 207–227. http://dx.doi.org/10.1145/322186.322187, Place: New York, NY, USA Publisher: Association for Computing Machinery.

Zeynali, A., Sun, B., Hajiesmaili, M., & Wierman, A. (2021). Data-driven competitive algorithms for online knapsack and set cover. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(12), 10833–10841. http://dx.doi.org/10.1609/aaai.v35i12.17294, URL: https://ojs.aaai.org/index.php/AAAI/article/view/17294. Number: 12.

Zhao, H., She, Q., Zhu, C., Yang, Y., & Xu, K. (2021). Online 3D bin packing with constrained deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(1), 741–749. http://dx.doi.org/10.1609/aaai.v35i1.16155, URL: https://ojs.aaai.org/index.php/AAAI/article/view/16155. Number: 1.