Pattern based learning through pricing for the bin packing problem

Abstract

Pattern is a popular form of knowledge and experience. Discovering and reusing high quality patterns has long been a main task in many data mining applications. In practical bin packing problems, mostly NP-Hard, the size of the next item to be packed can be unknown in advance. Many existing methods tend to reuse some example packing patterns that worked well in previous experiences. However, when the problem conditions (e.g. distributions of item sizes) change, the patterns that performed well for the previous circumstances can become less effective and adoption of these patterns would lead to poor sub-optimal solutions. It is, therefore, crucial to establish clear relationships between the problem solving circumstances and the quality levels of various patterns. Inspired by the dualism and the concept of shadow price in integer programming, in this research, we propose an iterative scheme to accurately quantify the value of patterns under each particular condition and then dynamically generate the "best" patterns based on the latest forecast of the problem condition. Our simulation results show that the proposed algorithm significantly outperforms existing heuristic methods when tackling online bin packing with finite discrete item types. The proposed method is generalisable for several other online combinatorial optimisation problems with linear/integer programming formulations.

1 Introduction

Combinatorial optimisation problems (COP) have extensive real-life applications. However, most of them are NP-Hard and finding the optimal solutions is normally computationally prohibitive for large-size instances. The problems become even harder when uncertainties are taken into account to improve the practicality of the solutions. The existing approaches to tackle these types of problems can broadly be classified into analytical model driven methods (typified by mathematical programming methods) and data-driven methods (e.g. genetic programming and reinforcement learning) [Bai *et al.*, 2022]. The former methods focus on the analytical properties of the mathematical model but may suffer from the robustness issues over uncertainties from the input data. The data driven methods often formulate the combinatorial problems as online optimisation problems and try to tackle the problem sequentially based on some policies or rules upon the realisation of random variables and the states of the partial solution at each decision point. One of the main drawbacks of the data driven methods is their inability to efficiently exploit the core structures and properties of the problem. More specifically, existing data driven methods primarily focus on the objectives to be optimised but often neglect various complex inter-dependencies among the decision variables (in the form of constraints) and their collective influence on the objective.

Patterns are one of the most powerful and effective problem solving tactics used in computer vision and big data analytics. It is also being used widely in solving combinatorial optimisation problems like the bin packing and vehicle routing. Patterns can be in diverse forms in different fields. Patterns as a problem solving strategy has several advantages: First, patterns are interpretable, editable and reusable. Therefore, the solutions built from patterns are more likely be accepted in practice. Second, patterns allow complex (including nonlinear) constraints to be implicitly modelled without breaking the preferred properties (e.g. linearity) of problem mathematical model. For example, in Amazon's last-mile vehicle routing competition [MIT, 2021], drivers' duty obligations and preferences are embedded in vehicle route patterns implicitly. Finally, patterns can be considered as a form of knowledge or experience-originated rules that can be analysed and polished to build better understanding about the problems and their solutions. However, for many combinatorial optimisation problems with uncertainties, we argue that the identification of good patterns alone is not sufficient to solve the problems. When the problem input data carry certain degree of uncertainties, patterns that are deemed "good" for some instances could become poor in quality in a different problem solving scenario caused by the uncertainties. The underlining causes are that, under uncertainties, although the problem structure (in terms of the objective function and the constraint structures) remain unchanged from instance to instance, the inter-dependencies between the decision variables may have changed significantly, leading to performance drop in some of the patterns. We shall illustrate this in the next section through a numerical example.

2 Preliminaries

2.1 Bin packing problem (BPP)

The bin packing problem is to use the minimum number of bins of identical sizes to pack a set of items that are in different sizes. In its basic offline version, the sizes of the items are given prior to the packing. Let B denote the capacity of the bins to be used and n be the number of item types, with each item type i having a size s_i and quantity q_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 by

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

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

$$\sum_{i=1}^{n} s_i x_{ij} \leq B y_j \quad \text{for } j = 1, \cdots, U \tag{3}$$

where U is the maximum number of possible bins to use. Bin packing problem is proved NP-Hard. To guarantee the computational efficiency, heuristic and meta-heuristics are often used. The most well-known heuristics include `best fit` (BF) and `minimum bin slack` (MBS) [Fleszar and Hindi, 2002] heuristics.

2.2 Online bin packing problem

Although most research efforts on BPP have been focusing on its `offline` version in which details of items to be packed are perfectly predictable in advance, many real-life packing problems appear in `online` because of dynamic realisation of items' specifications. More specifically, in model (1)-(3), the quantity of item type i, q_i, is not unknown but its proportion among all item types are known. Items arrive dynamically over time and its information (i.e. item type) is only available after arrival. A solution method for online BPP must assign a bin to each randomly arrived items upon its arrival and this assignment cannot be subsequently altered. Therefore, best fit remains a good solution method for online BPP with a competitive ratio of 1.7 but MBS is not usable anymore because it relies on the full information of items to be packed. In this research, we aim to improve average performance for online BBP by solving it with dynamically generated patterns through `pricing`. The motivation and underlining ideas can be illustrated by two simple BPP problems given Table 1 with bin capacity $B = 10$. For both problem cases, Best Fit gives sub-optimal solutions. Like most heuristic methods, Best Fit is a typical objective-focused incremental method that aims to obtain the maximum possible benefits in terms of objective (eq.(1)) at every step. However, although it partially addressed constraint (3) by using a simple rule to seek the best possible packing, the best-fit fails to address constraint eq.(2)) completely because it does not proactively considers the quantities of each item. Indeed, this is one of the main problems of many existing learning based approaches (e.g. genetic programming (DP) [Kucukyilmaz and Kiziloz, 2018] and deep reinforcement learning (DRL) [Vaswani *et al.*, 2017].

Table 1: Two simple 1D BPPs. s_i is item type size and $B = 10$.

case 1	$\{s_i\} = \{5, 4, 4, 3, 2, 2\}$ best-fit solution: $\{5, 4\}, \{4, 3, 2\}, \{2\}$ opt. solution: $\{5, 3, 2\}, \{4, 4, 2\}$
case 2	$\{s_i\} = \{5, 4, 4, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2\}$ best-fit solution: $\{5, 4\}, \{4, 3, 3\}$, $\{3, 3, 3\}, \{2, 2, 2, 2, 2\}, \{2\}$ opt. solution: $\{5, 3, 2\}, \{4, 3, 3\}$, $\{4, 3, 3\}, \{2, 2, 2, 2, 2\}$

In the two problem instances in Table 1, it can be seen that if the algorithms (e.g. GP or DRL) try to learn some policies directly from the solution in case 1 and use them to solve case 2, it would mostly fail because the packing patterns in the optimal solutions change significantly. Only one packing pattern (i.e. $\{5, 3, 2\}$) is reused. The optimal packing pattern $\{4, 4, 2\}$ disappeared completely and two new packing sets are now introduced for case 2. This means that if an algorithm tries to learn from good/optimal packing sets of some existing problem instances, it would most probably fail because the previously good packing sets may not be good any more for the current instance while previously unpopular patterns become more valuable.

Although the two instances are similar in terms of problem structure (e.g. bin size, item types), what is different is the distribution of item types. In case 2, more small items are to be packed. Therefore, the algorithm would need to not only look at how well each packing set performs (judged by the remaining capacity), but also consider how efficient it packs items that appear more frequently. Therefore, a possible improvement is to simultaneously consider both the objective of a combinatorial optimisation problem and its ability to fulfil the requirements of the problem (e.g. constraints). In some of the implementations of the DRL for COP, although satisfactability of constraints are often translated as part of state features, the algorithm was not provided with explicit measurements to indicate the quality of patterns with regard to the satisfaction of constraints.

In this paper, we propose to use the dualism of the COP to explicitly quantify the performance of different solution components (e.g. packing patterns for BPP) and use this information to guide the training of the algorithm to build the solution. The way that we quantify the quality of solution building blocks (e.g. packing patterns) is through the pricing in the dualism theory. The building block, or patterns, are dynamically generated by taking into account both the objective and constraints (2), (3) through pricing. We describe this more in detail in the next section.

2.3 Pricing and Duality

Duality is the principle that an optimisation problem could be viewed as two problems: the primal problem and dual problem. Consider the following standard optimisation problem formulation (denoted as `primal problem`) (4)-(6) [Boyd and Vandenberghe, 2004], where f is objective function, u and v are constraints:

$$\begin{aligned} \textbf{primal:} \quad & \text{minimise} & f(\mathbf{x}) & & (4)\\ & \text{subject to:} & u_i(\mathbf{x}) \geq 0 & \quad i = 1, \cdots, m & (5)\\ & & v_j(\mathbf{x}) = 0 & \quad j = 1, \cdots, n & (6) \end{aligned}$$

We can then write the associated Lagrange function $L(\mathbf{x}, \lambda, \eta) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i u_i(\mathbf{x}) + \sum_{j=1}^{n} \eta_j v_j(\mathbf{x})$, where λ and η are non-negative Lagrange multipliers. It is clear that finding an $\mathbf{x}^*$ that minimises $L(\mathbf{x}, \lambda, \eta)$ with proper (λ, η) set can also get the optimal solution of primal problem. We denote $g(\lambda, \eta) = \inf_{\mathbf{x}} L(\mathbf{x}, \lambda, \eta)$ be the `dual problem` that aims to find a lower bound of the primal problem. Maximising the dual problem will obtain a set of Lagrange multiplier (λ^*, η^*) that identify the effects that a certain constraint will have on the objective value. Such a value is also called `shadow price` in economy and management community. We extend the term price to describe the process of evaluating key components in a candidate solution. Although the dual problem is also intractable computationally for most COPs, it becomes solvable when f, u, v are linear functions, which is the case for the relaxed versions of many packing problems.

In offline combinatorial optimisation, there is rich literature that applies dualism and pricing to solve large-scale mixed integer programming problems in a branch-and-price framework [Barnhart *et al.*, 1998], which is essentially an iterative procedure to repeatedly solve a restricted master problem (RMP) and a pricing problem either exactly (e.g. [Wang *et al.*, 2010]) or heuristically (e.g. [Xue *et al.*, 2021]). In this paper, our goal is to extend the framework for the online bin packing problem.

3 Literature Review

A lot of real-world problems have close connections to bin packing problem. Therefore, it is probably one of the most studied combinatorial optimisation problems. A significant proportion of works focus on approximation algorithms with provable guarantees on the gap to the optimal solution. The most common ones are rule-based algorithms which could deal with both online and offline BPP problems. We use the term ”open/close” to indicate whether a bin is eligible to accept future items or not. [Johnson, 1973] firstly studied several online approximation algorithms which are essentially rules of thumb for bin assignments. The competitive ratio of these approximation algorithms for online BPP is proven to be around 1.7 although their average performances are often much higher. Improved heuristics were proposed later on by grouping items into subsets based on their sizes and then packing each subset of items by dedicated bins. Examples include Harmonic$_M$ [Lee and Lee, 1985] and Refined First Fit [Yao, 1980] heuristics.

Some of offline BPPs could be solved by exploiting the structural properties of BPP's integer programming formulations via exact algorithms like branch-and-bound schema [Martello, 1990] [Gilmore and Gomory, 1963] and branch-and-price methods [Nitsche *et al.*, 1999; Caprara and Monaci, 2009] but the computational time can vary significantly between instances.

Another strand of research efforts is the data-driven based methods that could exploit the distributional information of the random variables from the training data. Representative works include the use of genetic programming based hyper-heuristic to train a packing strategy/policy [Burke *et al.*, 2006], and the evolutionary algorithms for evolving rules to select the most appropriate packing heuristics at each decision point [Ross *et al.*, 2003; López-Camacho *et al.*, 2014].

Deep reinforcement learning (DRL) has gained growing attention in combinatorial optimisation, including BPP. In most cases, BPP was formulated as Markov Decision Process (MDP) through which uncertainties can be effectively handled. [Zhang *et al.*, 2021; Zhao *et al.*, 2021] proposed reinforcement learning based methods to tackle online 3D BPP by an end-to-end manner. For 1D bin packing problem, [Hubbs *et al.*, 2020] established a set of environments for classical OR problems and [Balaji *et al.*, 2019] introduced a set of DRL benchmarks.

4 Methodology

In this section, we argue that standard deep reinforcement learning without sufficiently exploiting BPP constraints may be either inefficient in training or inferior in solution quality. This is because constraints can have as much impact on the optimisation as that the objective function has. We show that dynamic pattern discovery through dual pricing could handle both the objective and constraints very well and lead to significant performance improvement for online BPP. This use of patterns as a general online COP solving strategy is beneficial in many aspects.

In our online BPP, we assume a finite number of items and a known distribution of item types $\rho \sim D$. A problem instance is defined as a sequence items, with the type of each item being sampled from D. In reality, 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 problem which is harder to solve. We investigated both problem cases in this research.

We propose a general framework (see Figure 1) that adopts explicitly the dualism of COP to assist pattern based solution building.

4.1 Knowledge of distribution

With a priori knowledge of a problem, better packing strategies than the heuristics without using such knowledge could be built. In this work, the knowledge of the distribution of item sizes is used. Probability mass function is applied in BPPs with standard distributions, uniform and normal distribution for example. In the complex cases where the distribution is hard to describe mathematically, a data-driven method is adopted to retrieve it from historical data .

In this work, a Gaussian Mixture Model (GMM) is adopted to learn the knowledge of distribution. The model is made up with several normal distributions with different means $\mu = [\mu_0, ...\mu_s]$ and variances $\sigma = [\sigma_0, ...\sigma_s]$ and the probability is computed by averaging those distributions. The model is trained with historical data to gain a priori distribution.

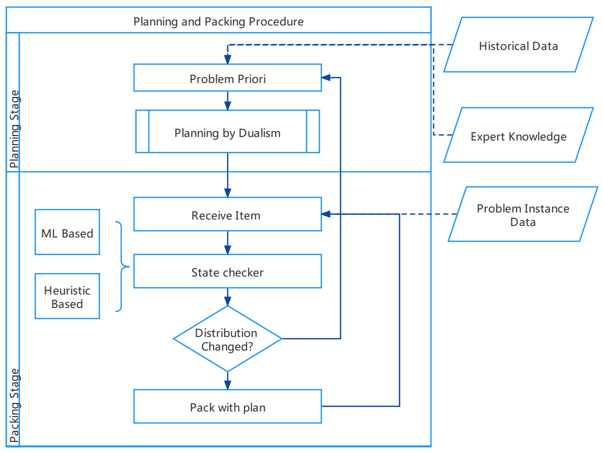

Figure 1: The proposed pattern based packing algorithm. Given a state in the whole problem, We first use the duality of algorithm to generate a set of patterns, then use the pattern set to build problem solving strategy for an online problem. A state monitor module is used to check whether we should re-generate the pattern set according to the new state.

4.2 Patterns and Plan for Packing

We define a BPP `pattern` (shown in Figure 2 and Table 1) as a combination of different items that fills in a bin. A pattern can be used multiple times in a solution. Let P be the set of all feasible packing patterns for a bin of size B and z_p the frequency that pattern $p \in P$ is used in a solution. The original BPP formulation can be re-formulated as follows:

$$\text{minimise} \quad \sum_{p\in P} z_p \tag{7}$$

$$\text{subject to} \quad \sum_{p\in P} c_p^i z_p \geq q_i, \quad \forall i \tag{8}$$

where c_p^i denotes how many times item type i appears in pattern p. P can be generated by

$$P = \{p | \sum_i c_p^i s_i \leq B, c_p^i \in \mathbb{N}^0\} \tag{9}$$

In practice, it is not possible to enumerate the entire set of P. The optimisation often starts with a small subset of P and the corresponding problem is called a Restricted Master Problem (RMP).

We define a packing `plan` as a set of (p, z_p) pairs where z_p is the frequency that p is used in the plan. For the problem case 2 in Table 1, for example, the packing plan can be represented by $\{(\{5,3,2\},1), (\{4,3,3\},2), (\{2,2,2,2,2\},1)\}$.

In online BPP, demand of each item type, q_i, is an estimate value which may deviate slightly from the distribution used to generated a plan. The plans need to be adjusted when the actual realisation of the random variables are different from the estimate values. An adaptive packing strategy is proposed, as shown in Algorithm 1, to adapt to the estimation error. With the algorithm, the items are packed according to the plan. A demand satisfaction table is used to keep record of the ratio of fulfilled demands to their expected demands for each item type. Whenever the satisfaction ratio of a particular item

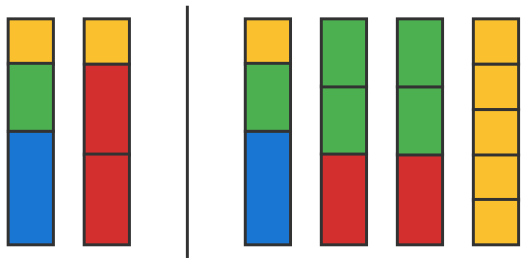

Figure 2: Patterns used in the optimal solutions for cases in Table 1. Different item sizes are illustrated by different colours. The patterns on the left side of the vertical bar are for problem case 1 and patterns on the right side are for case 2. Only one pattern appears in both cases. Blue: size 5, red: size 4, blue: size 3, yellow: size 2.

type exceeds 100%, the corresponding item is considered unplanned. When this happens, the item is assigned with high priority to the opened bin by Best Fit. If this is not possible, a re-planning is triggered with the updated demands by resolving problem formulation (7)-(8). The process repeats until all items are packed.

Algorithm 1 Algorithm of packing strategy

```
1: Input: item sequence $L$, initial estimated distribution $D_0$, look-back step $k$
2: Generate plan $\mathcal{P}$ by solving formulation (7)-(8) with the expected demand $q_i$.
3: Setup demand satisfaction table $d$, opened bin list $B$
4: for $l \in L$ do
5:     $\bar{D} \leftarrow \texttt{estimate\_distribution}(l-k, ...l-1)$
6:     $D_l \leftarrow \texttt{state\_checker}(D_{l-1}, \bar{D})$, estimate $d_i = \mathbb{E}[D_l^i]$
7:     $\mathcal{P} \leftarrow \texttt{replan}(d)$
8:     if $i$ can pack according to plan then
9:         $\texttt{pack\_item}(l, B, \mathcal{P})$
10:    else
11:        if Item can insert to previous bin regardless pattern then
12:            $\texttt{best\_fit\_pack}(l, B)$
13:        else
14:            $\mathcal{P} \leftarrow \texttt{replan}(d)$
15:            $\texttt{pack\_item}(l, B, \mathcal{P})$
16:        end if
17:    end if
18:    Update demand table $d$
19: end for
```

4.3 Pattern and Plan Generation

We propose an iterative two-stage algorithm with planning and packing (see Figure 1). Starting from an initial plan, the items are packed according to the plan iteratively. When an item cannot not be packed into any of the current open bins, planing is triggered again. The 'prices' of all existing patterns and new patterns will be re-evaluated according to the updated demand estimation (i.e. new estimations of unpacked items). In this way, our algorithm is able to handle small estimation errors. More importantly, it also adapts to new distributions of the random variables, as confirmed by the computational results in Section 5.2.

When the model (7)-(8) is solved, a key step is to iteratively find a good pattern to add into the pattern set P so that we don't need to enumerate all feasible patterns. To achieve this, a sub-problem (called pricing problem or pattern generation) is repeatedly solved to find a pattern $p*$ that minimises the reduced cost function (eq. (10)) for given shadow prices δ_i associated with demand constraint (8), while satisfying the packing constraint (11) at the same time.

$$\text{minimise} \quad 1 - \sum_i \delta_i c^i_{p*} \tag{10}$$

$$\text{subject to} \quad \sum_i s_i c^i_{p*} \leq B \tag{11}$$

The pattern generation process stops once the objective value of eq. (10) becomes non-negative which indicates that all potential cost-reducing patterns have been successfully discovered. The problem (7)-(8) can be solved by using a commercial integer programming solver.

4.4 State checker and re-planning

When the distribution is unbalanced over time, i.e. item are sampled from different distributions periodically, the estimation of a priori distribution is unlikely to be accurate. We propose a state checker module to check the shift of distribution and change the plan accordingly. It traces the past k steps to estimate the current distribution, as illustrated in Algorithm 1 line 6. The similarity between a priori distribution D_l and current distribution $\bar{D}$ is calculated by Kullback-Leibler divergence $KL(D_l||\bar{D})$. If it is greater than the predefined threshold θ, it means that a priori distribution is not accurate, and therefore the plan needs to be updated. If the possible distributions are finite and known, we just switch to the one with the lowest KL-divergence from the current distribution. For complex distributions, the learning model is fine-tuned according to observations. After we identify and update the distribution, the plan will be updated accordingly.

5 Experiments

We setup three experiments for three types of distributions, including a stationary distribution, a non-stationary distribution and a dual distribution. The first two experiments are set up by sampling items from predefined distributions. The third experiment adopts the dataset in [Burke *et al.*, 2010] in which items were sampled from two different normal distributions. For all three experiments, the item sizes are integers less than or equal to 100, and the capacity of bin is set to be 100. Ranking test is used to measure the performance between our method and Best Fit by counting the winning rate of our methods over all instances. Additionally, the solutions are also compared with a recent DRL method [Balaji *et al.*, 2019] and the optimal solution of the offline version of the problem instance computed by the Gurobi MIP solver.

5.1 Stationary Distribution

In this experiment, 3 set of problems are generated, corresponding to 3 distributions: uniform, binomial with $p = 0.5$ and non-trivial discrete, respectively. Each set contains 100 instances and each instance has 1000 items. For each test set, we set up the item type to be $\{10, 15, ..., 55\}$, in total 10 types. The number of each type in set1 subject to uniform distribution, and set2 subject to normal distribution. We set up the non-trivial probability of each type in set3 to be: $\{0.2, 0.05, 0.1, 0.12, 0.08, 0.05, 0.03, 0.01, 0.18, 0.18\}$.

The distribution of different item types is used in the algorithm while their exact quantities are not used. The first part of Table 2 shows the experiment results of winning rate and average number of bins gap from optimal solution. It can be seen that out of 100 instances in each set, the win rates of our method against Best Fit range from 61% to 100%. Our methods also achieves better average gap than Best Fit and DRL.

Table 2: Stationary and non-stationary distribution test results

Id	Win rate vs. BF	Avg. Gap from Opt.		
		Ours	DRL	BF
set1	75%	2.7	3.5	3.8
set2	100%	4.9	15.1	15.3
set3	61%	4.4	4.9	5.0
set4	95%	8.0	25.8	16.0
set5	91%	7.9	17.8	14.4
set6	79%	8.3	17.2	12.4

5.2 Non-stationary Distribution

In this experiment, we generate 3 sets of problemss (Set4-6) using 3 different periodic non-stationary distributions with item types shown in the Section 5.1. Set4 is given by switching between distributions 1 and 2. Set5 is given by switching between distributions 1 and 3. Set6 is given by distribution switches of 1,2,3. In each set, we run 100 sampled instances, each of which has 3000 items. The distribution changes to the next after every 300 items. The second part of Table 2 gives non-stationary test results. Our method also outperforms Best Fit in most cases of this experiment.

5.3 Dual Distribution

The aim of this experiment is to test the performance of our method in an environment that the distribution of items is not easy to acquire via statistic. The data is given in [Burke *et al.*, 2010], which combined two data sets of normal distributions. There are 8 experiment sets, each has 20 instances with 5000 unsorted items. The 2-component Gaussian Mixed Model (GMM) trained with 10000 randomly sampled data is used.

Experimental results are given in Table 3. In the second column, the values in bold indicate the cases that our model outperforms Best Fit. Our method significantly outperforms Best Fit in sets 7-10 while the solution quality for sets 11-14 (Multinomial type 2 by [Burke *et al.*, 2010]) is not very competitive. This indicates a possible weakness of the proposed pattern based method when handling highly dynamic and volatile data. The algorithms that focus more on short-term gains such as Best Fit perform well in that situation.

Table 3: Dual distribution results

Id	Win rate vs. BF	Avg. Gap from Opt.	
		Ours	BF
set7	**100%**	**30.0**	131.4
set8	**100%**	**29.7**	120.9
set9	**80%**	**31.3**	45.7
set10	**70%**	**25.1**	26.3
set11	65%	64.5	49.0
set12	**75%**	**52.6**	73.8
set13	45%	49.0	47.0
set14	35%	26.2	19.3

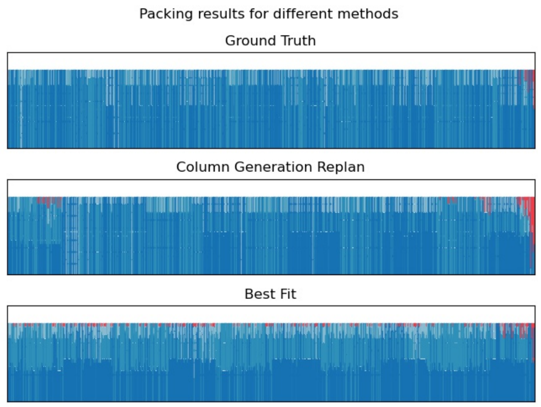

Figure 3: Solution visualisation of two methods and ground truth. Blue bars represent the items. The darker an item is, the greater its size is. Red bars indicate waste space.

5.4 Analysis of experiment results

We have three major conclusions from the experiment results:

1. **Pattern selection is important.** Figure 3 visualises the solutions of a problem instance. The ground truth has nearly all bins filled. Our model makes most bins filled but still has a few bins with significant empty spaces used as buffers for forthcoming items. On the contrary, Best Fit tends to leave many bins with small waste spaces which cannot be filled by any item.
2. **Dynamic pattern prices work well.** The performance of the proposed method on non-stationary distribution instances indicates the benefits of dynamic adjustment of pattern prices in handling the changes of distributions. In contrast, the Best Fit heuristic cannot utilise this information.
3. **Distribution Estimation Quality Critical.** Our method works extremely well when the distribution of random variables is estimated reliably. However, if the estimation quality is low (see Figure 4), the performance may deteriorate noticeably, evidenced by the inferior results for sets 11-14 in Table 3. When the distribution of items varies and it is difficult to predict the changes, our pattern based method will be much more affected than simple heuristics like Best Fit.
4. **Comparison with DRL** We also compared our method with DRL benchmark from [Balaji *et al.*, 2019] on sets 1-6. Each DRL model was trained with 1000 randomly sampled instances with same distribution of items. In terms of solution quality, the proposed method outperformed both DRL and Best Fit significantly. DRL outperformed Best Fit with an average of 0.2 bins for stationary solutions while it did not perform well in non-stationary cases. A possible advantage of DRL is its relatively low inference time once they are trained. The most time consuming procedure in the proposed method is re-planning. For example, the average times of re-planning was *3.1* in the experiments of sets 1-6, which was similar to the time cost of solution generation for DRL.

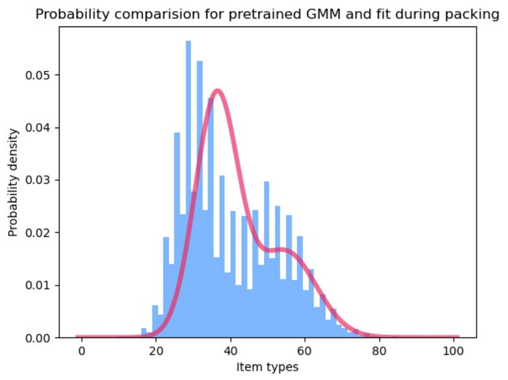

Figure 4: Visualisation of the GMM results The histogram represents actual item size counts, red curve represents the estimated probability density by GMM.

6 Conclusion and Future works

In combinatorial optimisation, patterns are reusable building blocks of solutions that are more favourable than black-box solvers. However, we showed in this research that the values of patterns could change due to uncertainties related to objectives and constraints and most existing methods fail to exploit the inter-dependencies among decision variables incurred by uncertainties in constraints. To address this, we established a scheme to dynamically quantify the usefulness of different patterns based on the dualism of COP and use the information to guide the decision process. A state checker is used to monitor possible switches of distributions of the randomness. The test results on bin packing problem show significant performance advantage of the proposed method for instances with small and medium levels of distribution errors compared with the existing state of the art methods. In future, we would investigate other novel ways to combine learning methods with patterns and dynamic pricing.

References

[Bai *et al.*, 2022] Ruibin Bai, Xinan Chen, Zhi-Long Chen, Tianxiang Cui, Shuhui Gong, Wentao He, Xiaoping Jiang, Huan Jin, Jiahuan Jin, Graham Kendall, et al. Analytics and machine learning in vehicle routing research. *Inter-*

national Journal of Production Research, pages in–press, 2022.

[Balaji *et al.*, 2019] Bharathan Balaji, Jordan Bell-Masterson, Enes Bilgin, Andreas C. Damianou, Pablo Moreno Garcia, Arpit Jain, Runfei Luo, Alvaro Maggiar, Balakrishnan Narayanaswamy, and Chun Ye. ORL: reinforcement learning benchmarks for online stochastic optimization problems. *CoRR*, abs/1911.10641, 2019.

[Barnhart *et al.*, 1998] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, 46(3):316–329, June 1998.

[Boyd and Vandenberghe, 2004] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK ; New York, 2004.

[Burke *et al.*, 2006] 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.

[Burke *et al.*, 2010] Edmund K. Burke, Matthew R. Hyde, and Graham Kendall. Providing a memory mechanism to enhance the evolutionary design of heuristics. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.

[Caprara and Monaci, 2009] Alberto Caprara and Michele Monaci. Bidimensional packing by bilinear programming. *Mathematical Programming*, 118(1):75–108, April 2009.

[Fleszar and Hindi, 2002] K. Fleszar and K.S. Hindi. New heuristics for one-dimensional bin-packing. *Computers and Operations Research*, 29:821–839, 2002.

[Gilmore and Gomory, 1963] P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting Stock Problem—Part II. *Operations Research*, 11(6):863–888, December 1963.

[Hubbs *et al.*, 2020] Christian D. Hubbs, Hector D. Perez, Owais Sarwar, Nikolaos V. Sahinidis, Ignacio E. Grossmann, and John M. Wassick. Or-gym: A reinforcement learning library for operations research problem. *CoRR*, abs/2008.06319, 2020.

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

[Kucukyilmaz and Kiziloz, 2018] Tayfun Kucukyilmaz and Hakan Ezgi Kiziloz. Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem. *Computers & Industrial Engineering*, 125:157–170, November 2018.

[Lee and Lee, 1985] C. C. Lee and D. T. Lee. A simple on-line bin-packing algorithm. *Journal of the ACM*, 32(3):562–572, July 1985.

[López-Camacho *et al.*, 2014] Eunice López-Camacho, Hugo Terashima-Marin, Peter Ross, and Gabriela Ochoa. A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications*, 41(15):6876–6889, November 2014.

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

[MIT, 2021] MIT. Amazon last mile routing research challenge. https://routingchallenge.mit.edu/, 2021.

[Nitsche *et al.*, 1999] Christoph Nitsche, Guntram Scheithauer, and Johannes Terno. Tighter relaxations for the cutting stock problem. *European Journal of Operational Research*, 112(3):654–663, February 1999.

[Ross *et al.*, 2003] Peter Ross, Javier G. Marín-Blázquez, Sonia Schulenburg, and Emma Hart. Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics. In *Genetic and Evolutionary Computation — GECCO 2003*, pages 1295–1306, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[Wang *et al.*, 2010] Congcong Wang, Peter B. Luh, Paul Gribik, Li Zhang, and Tengshun Peng. The subgradient-simplex based cutting plane method for convex hull pricing. In *IEEE PES General Meeting*, pages 1–8, July 2010.

[Xue *et al.*, 2021] Ning Xue, Ruibin Bai, Rong Qu, and Uwe Aickelin. A hybrid pricing and cutting approach for the multi-shift full truckload vehicle routing problem. *European Journal of Operational Research*, 292(2):500–514, July 2021.

[Yao, 1980] Andrew Chi-Chih Yao. New Algorithms for Bin Packing. *Journal of the ACM*, 27(2):207–227, April 1980.

[Zhang *et al.*, 2021] Jingwei Zhang, Bin Zi, and Xiaoyu Ge. Attend2Pack: Bin Packing through Deep Reinforcement Learning with Attention. *arXiv:2107.04333 [cs]*, August 2021.

[Zhao *et al.*, 2021] Hang Zhao, Qijin She, Chenyang Zhu, Yin Yang, and Kai Xu. Online 3D Bin Packing with Constrained Deep Reinforcement Learning. *Nature*, March 2021.