Randomized Local Search for Two-Dimensional Bin Packing and a Negative Result for Frequency Fitness Assignment

Rui Zhao[1][a], Zhize Wu[1][b], Daan van den Berg[2][c], Matthias Thürer[3][d], Tianyu Liang[1][e], Ming Tan[1][f] and Thomas Weise[1][g]

[1]*Institute of Applied Optimization, School of Artificial Intelligence and Big Data, Hefei University, Jinxiu Dadao 99, Hefei, 230601, Anhui, China*

[2]*Department of Computer Science, Vrije Universiteit Amsterdam, De Boelelaan 1111, Amsterdam, 1081 HV, The Netherlands*

[3]*Professur Fabrikplanung und Intralogistik, TU Chemnitz, Str. der Nationen 62, 09111 Chemnitz, Sachsen, Germany*

Keywords: Two-Dimensional Bin Packing, Randomized Local Search, Frequency Fitness Assignment, Cutting Stock Problem.

Abstract: We consider a two-dimensional orthogonal bin packing problem (2BP) where rectangular items are to be placed into rectangular bins such that their edges are parallel to those of the bins with the aim to require as few bins as possible. Two variants of the problem are analyzed. In the 2BP|O|F, the items have a fixed orientation while in the 2BP|R|F, they can be rotated by 90 degrees. We show that on both variants, a simple randomized local search (RLS) has surprisingly good performance – if the objective function guiding the search is defined suitably. In particular, on the 2BP|O|F, the RLS performs on par with more complicated state-of-the-art metaheuristics. We furthermore investigate plugging Frequency Fitness Assignment (FFA) into the RLS, obtaining the FRLS. FFA has improved the RLS performance on several classical $\mathcal{NP}$-hard optimization problems from operations research, including Max-SAT, the Job Shop Scheduling Problem, and the Traveling Salesperson Problem. This paper is the first negative result for FFA: it cannot improve algorithm performance on the 2BP variants studied. This can be explained by the fact that RLS already performs very well on the instances of the 2DPackLib benchmark set used as the basis of our experiments.

1 INTRODUCTION

Cutting stock problems (CSPs) and bin packing problems (BPs) are two closely related domains of operations research (Lodi et al., 2002; Iori et al., 2021). CSPs ask for dividing larger chunks of material into smaller pieces, whereas BPs require us to place smaller items into larger containers. In many cases, variants of both problems can be trivially transformed into each other. Their two-dimensional orthogonal variants, which we here jointly refer to as 2BPs, have many important applications, ranging from obvious tasks such as packing and layout to scheduling and build formation in additive manufacturing (Pinto et al., 2024; Li and Zhang, 2018). The goal of solving a 2BP is to pack a set of rectangular items into as few as possible rectangular bins. While several exact algorithms for this purpose have been developed (Ma and Zhou, 2017; Cid-Garcia and Rios-Solis, 2020; van den Berg et al., 2016; Braam and van den Berg, 2022; Martello and Vigo, 1998; Iori et al., 2021), they can only be one part of the answer to the 2BP due to its $\mathcal{NP}$-hard nature (Lodi et al., 2002). As a result, several heuristic algorithms have been applied to this problem family, ranging from one-shot constructive heuristics (Wong and Lee, 2009; Liu and Teng, 1999; Pejic and van den Berg, 2020) over tabu search (Lodi et al., 2004), evolutionary (Kierkosz and Luczak, 2013; Liu and Teng, 1999; Gonçalves and Resende, 2013; Lee, 2008; Li et al., 2021), and memetic algorithms (Blum and Schmid, 2013; Parreño et al., 2010)

[a] https://orcid.org/0009-0002-2426-9453
[b] https://orcid.org/0000-0001-7416-5711
[c] https://orcid.org/0000-0001-5060-3342
[d] https://orcid.org/0000-0002-2705-969X
[e] https://orcid.org/0009-0004-3732-4831
[f] https://orcid.org/0009-0004-3851-053X
[g] https://orcid.org/0000-0002-9687-8509

Zhao, R., Wu, Z., van den Berg, D., Thürer, M., Liang, T., Tan, M. and Weise, T.
Randomized Local Search for Two-Dimensional Bin Packing and a Negative Result for Frequency Fitness Assignment.
DOI: 10.5220/0012888500003837

In *Proceedings of the 16th International Joint Conference on Computational Intelligence (IJCCI 2024)*, pages 15-26
ISBN: 978-989-758-721-4; ISSN: 2184-3236

to hyper-heuristics (Beyaz et al., 2015; Terashima-Marín et al., 2007).

We build on the recent work (Zhao et al., 2024), where it was shown that a randomized local search (RLS) can perform surprisingly well on a variant of the 2BP where item rotation is permitted, if the objective function is defined appropriately. The **first contribution** of this paper is to significantly expand upon these results and to show that the same RLS can be even more competitive to the related work on the 2BP if item rotation is not permitted. In other words, we find that two important variants of the 2BP can already be solved quite well with relatively simple algorithms.

As our related work study in Section 3 shows, many of the existing works on the 2BP provide results that are either normalized with different lower bounds, are always averaged over several instances, or use different benchmark instances. The **second contribution** of our work therefore is to provide a complete set of results on the recently published 2DPackLib benchmark (Iori et al., 2022), including the discovered packings, the complete progress over the runtime, and the algorithm implementations in an immutable archive at https://doi.org/10.5281/zenodo.13324219, to serve as basis for future research.

All of the metaheuristic algorithms for the 2BP share the common concept that they iteratively sample new solutions s_n based on the currently retained solutions s_c and that they tend to retain the new solution if it is better (or, at least, not worse) than s_c. They may maintain populations of solutions (e.g., evolutionary algorithms) or introduce some diversity or exploration criterion (e.g., tabu search), but over time, better solutions are preferred over worse ones. There are only three iterative optimization algorithms free of such bias: random sampling, random walks, and exhaustive enumeration.

In (Weise et al., 2014b), a fourth optimization approach without bias towards good solutions was introduced, Frequency Fitness Assignment (FFA), but its theoretical properties were proven only relatively recently (Weise et al., 2021b; Weise et al., 2023). FFA is not an algorithm itself, but a module that can be plugged into a wide range of iterative heuristics. It then renders them invariant under all injective transformations of the objective function value. It prefers new solutions s_n if they have a previously less frequently encountered objective value $z_n = f(s_n)$, regardless of whether they are better or worse.

Despite this unbiasedness, FFA yields remarkable performance on several classical $\mathcal{NP}$-hard optimization problems such as Max-SAT (Weise et al., 2021b; Weise et al., 2023), the Job Shop Scheduling Problem (JSSP) (Weise et al., 2021a; de Bruin et al., 2023), and on the Traveling Salesperson Problem (Liang et al., 2022; Liang et al., 2024, TSP). Experiments support that FFA-based algorithms tend to find better solutions than the objective-guided counterparts into which they are plugged if the problem they are applied to does not have too many different objective values (Liang et al., 2022; Liang et al., 2024).

Since the possible range of the number of bins into which the items in a 2BP can be packed is usually small (Zhao et al., 2024), the 2BP might be another classical $\mathcal{NP}$-hard problem where FFA could excel. As the **third contribution**, we explore this idea by plugging FFA into the RLS, yielding the FRLS, and applying it to all the 2DPackLib instances. We find that FRLS cannot outperform the RLS on the 2BP. This is the first completely negative result for FFA on any classical optimization task. However, when analyzing the performance of RLS and FRLS in more detail, the reasons for this discrepancy become clear.

The **fourth contribution** of our work is that we identify the need for harder benchmark instances based on reproducible results: RLS should not perform well on $\mathcal{NP}$-hard problems, but it does so anyway in our experiments on the 2BP. This means that we have confirmed that the 2DPackLib benchmark for the 2BP, despite being well-designed and comprehensive, is probably too easy and should be extended.

In the remainder of the paper, we first introduce the two variants of the 2BP in Section 2 and discuss the related work on it in Section 3. We present our approach to the 2BP, including the RLS and FRLS algorithms, the encoding used, the search space, operators, and objective functions, in Section 4. The experimental results are analyzed in Section 5 before we conclude the paper with a summary and an outlook for future work in Section 6.

2 2D BIN PACKING

In this paper, we consider two orthogonal rectangular two-dimensional bin packing problem variants, which are defined as follows. The bins and items are both two-dimensional rectangles. All bins have the same width $W \in \mathbb{N}$ and height $H \in \mathbb{N}$. The number of available bins is unlimited. There are $N \in \mathbb{N}$ items. The item $i \in \{1..N\}$ has width $w_i \in \{1..W\}$ and height $h_i \in \{1..H\}$. For each item i, there is a demand d_i, meaning that d_i instances of item i need to be packed. Thus, the goal is to pack *all* the $T = \sum_{i=1}^{N} d_i$ item instances into as few bins as possible. The edges of the item instances must be parallel to those of the bins. There must be no overlap and all item instances must be contained entirely in the bins.

In the 2BP|O|F variant, the items have a fixed orientation, must not be rotated, and are placed in the same way they are defined in the problem instance (Lodi et al., 2002)[1]. In the 2BP|R|F problem variant, the item instances can be rotated by 90°.

Both variants have packing plans s as solutions, which can be defined as sets of T records $s[j]$ with $j \in \{1..T\}$, each denoting the location of one packed item instance in its bin. Thus, record $s[j]$ stores the item $s[j].i \in \{1..N\}$, the bin $s[j].b \in \{1..T\}$ into which it should be packed, as well as the horizontal coordinate $s[j].x_1 \in \{0..W-1\}$ and the vertical coordinate $s[j].y_1 \in \{0..H-1\}$ of the lower left corner of the packed item instance. To represent whether the item instance is rotated by 90° or not, the coordinates $s[j].x_2 \in \{1..W\}$ and $s[j].y_2 \in \{1..H\}$ of its upper right corner are stored as well. For $s[j].i = i$, it then either holds that $s[j].x_2 = s[j].x_1 + w_i$ and $s[j].y_2 = s[j].y_1 + h_i$ if the instance of item i is not rotated or $s[j].x_2 = s[j].x_1 + h_i$ and $s[j].y_2 = s[j].y_1 + w_i$ if the item instance is rotated. Rotation is only permitted in the 2BP|R|F variant.

The space of all such feasible packing plans is $\mathbb{S}$. The goal is to find the packing plan $s \in \mathbb{S}$ that requires the fewest bins among all the possible feasible plans and, hence, also has the smallest total unoccupied space inside the bins, i.e., that minimizes the following objective function:

$$f_1(s) = |\{s[j].b \, \forall j \in \{1..T\}\}| \quad (1)$$

For any 2BP instance, it is not a priori clear how many bins will be required in the optimal solution. However, lower bounds for f_1 provide a limit for the best-case scenario. The geometric or continuous bound $lb_g(f_1)$ therefore returns the rounded-up quotient of the total area sum of all item instances and the bin area (Martello and Vigo, 1998):

$$lb_g(f_1) = \left\lceil \left(\sum_{i=1}^{N} w_i h_i d_i\right) / (WH) \right\rceil \quad (2)$$

Obviously $f_1(s) \geq lb_g(f_1)$ for all $s \in \mathbb{S}$, as it is impossible to package the items into bins whose total area is less than the total item area. The most commonly used lower bound for f_1 may be $lb_d(f_1)$ by (Dell'Amico et al., 2002), which, due to its algorithmic formulation, shall not be detailed here. In our work, we will use the maximum $lb_m(f_1) = \max\{lb_g(f_1), lb_d(f_1)\}$ of both bounds where appropriate.

[1]The "F" stands for cutting being free if the problem is considered from the CSP perspective.

3 RELATED WORK

A search for publications focused on these specific problem variants, in particular the 2BP|R|F, produces significantly fewer results (Cid-Garcia and Rios-Solis, 2020) compared to other classical problem domains such as the Traveling Salesperson Problem or the Quadratic Assignment Problem. This may be due to the many different problem variants, their sometimes non-obvious naming, and, finally, due to the comparatively higher implementation effort for simple algorithms (see Algorithm 3). Nevertheless, the orthogonal rectangular 2BP with and without rotation did attract some research attention over the decades, although maybe not as much as it deserves.

(Bengtsson, 1982) contributed a heuristic algorithm for packing rectangular pieces in 1982. The algorithm involves an initial allocation of items to bins and then tries to iteratively refine it. The goal of this algorithm is to minimize the unused space in the bins and it permits that some items may not be loaded, so the results are not directly comparable to our 2BP scenarios. The authors provide the beng benchmark and use it in their experiments, where they achieve a bin utilization between 95% and 98% within computational budgets of 0.5 and 1 second.

(Liu and Teng, 1999) proposed an Improved Bottom Left (IBL) encoding that can translate signed permutations to packings. In their work, the goal is to minimize the packing height in a bin of infinite height. Examples are provided, but no experimental results. However, we will adapt the IBL to the 2BP in this work (see Algorithm 3 later on).

(Lodi et al., 2004) developed the C library TSpack for solving two- and three-dimensional bin packing problems with the goal of minimizing the number of bins. It offers iterative optimization through Tabu Search whose key aspect is the ability to switch between neighborhoods of different sizes. The goal is to tune between intensification and diversification. The authors use a dataset similar to class, but with different instances and hence, different bounds and solutions. The results thus cannot be directly compared with such obtained on 2DPackLib. The computational budget per run is 60s.

(Terashima-Marín et al., 2007) introduce two hyperheuristic approaches to the 2BP|R|F. They first define a set of selection heuristics that choose items and bins as well as a set of placement heuristics that place the selected items into the selected bins. Then, they synthesize rules that decide which of the heuristics should be applied based on the current state of the packing process. For this purpose, they both investigate an XCS-type Learning Classifier System and a dedicated Genetic Algorithm (GA). The authors show that the hyper-heuristics can synthesize heuristics that can outperform the best single heuristic on any instance, however, no results are reported that could be used for direct comparison.

In the MultiCrossover GA (MXGA) by (Lee, 2008), solutions are integer strings storing, for each of the T item instances, into which bin it should be placed. The location of the item instances is then computed by a heuristic placement routine. As operators, single-point crossover and single-swap mutation are applied. The work again uses the class benchmark and grants 120 seconds per run to its C-based algorithm implementations. The number of repetitions is not given. Results are averaged over instance groups, divided by lb_d and given with a precision of two decimals.

(Wong and Lee, 2009) propose two heuristic placement algorithms, namely Improved Lowest Gap Fill (LGFi) for the case with rotation and LGFi_{OF} for the case where the items cannot be rotated. These algorithms iteratively select the bin with the smallest remaining space to place the current item until all items are placed. LGFi is a constructive heuristic that creates a single packing. It would therefore be a possible alternative to the IBL method that we use as the encoding scheme in our RLS and FRLS in Section 4. Results for the class instances are reported and normalized by lb_d and another bound (Boschetti and Mingozzi, 2003) with three decimals of precision.

(Parreño et al., 2010) developed the GRASP/VND algorithm for 2BP|O|F problems, which combines the greedy randomized adaptive Search Procedure (GRASP) and the variable neighborhood descent (VND). GRASP is a constructive algorithm that builds solutions incrementally by iteratively selecting the best available option based on randomized greedy criteria. It aims to balance exploitation (choosing the best immediate option) and exploration (diversifying the search space). In GRASP/VND, the GRASP procedure generates an initial solution by iteratively adding items to bins based on a randomized greedy rule. VND is a local search algorithm that explores different neighborhoods around a given solution to find local optima. It iteratively moves from one neighborhood to another until no further improvement is possible. In GRASP/VND, the VND is applied to the initial solution and improves it iteratively. The algorithm is run for 50000 iterations on the class and beng datasets. The average of the numbers of bins per instance group is reported.

(Gonçalves and Resende, 2013) present the Biased Random Key GA (BRKGA) for 2D and 3D bin packing problems with and without item rotation (BRKGA-2r, BRKGA-aNB, respectively). The chromosome encodes both the sequence in which items are packed as well as their orientation. The authors use the class and the beng instances, and also some other benchmarks to evaluate their algorithm. They conduct three runs per setup for 200 generations with a population of 30 times the number T of item instances, and report the average number of bins over the instance groups.

The EA-LGFi for the 2BP|O|F by (Blum and Schmid, 2013) is an Evolutionary Algorithm (EA) that works on permutations and uses the aforementioned LGFi heuristic by (Wong and Lee, 2009) to translate the permutations to packing plans. The paper used class instances as the benchmark and found four new best solutions. The computational budget is 10^6 FEs per run and one run is conducted per instance. The sum of bins over all the solutions per instance group is reported.

(Kierkosz and Luczak, 2013) developed an evolutionary algorithm (EA) to select a subset of the items and place them such that the maximum area in the single available bin is used. None of the benchmarks in 2DPackLib are used.

(Beyaz et al., 2015) introduced their hyper-heuristic method HHA-NO based on a Memetic Algorithm, i.e., the combination of an EA with local search. The genome contains the order of items to be packed as well as two heuristic selections. The first half of the items are packed using the first selected heuristic and the second half using the second selected heuristic. A population of 60 individuals evolved for 40 generations. The algorithm is implemented in C++ and the runtimes within the range of 46s to 14.5min are reported for a selection of instances. This runtime seems to be relatively high for what should be around $60*40 = 2400$ objective function evaluations (FEs), raising the question of how to fairly compare algorithms whose single steps require vastly different runtimes (Weise et al., 2014a). The authors use the class instances and report the *sum* of the f_1 values over the instance set.

(Ma and Zhou, 2017) introduced two mixed-integer programming (MIP) models for solving the 2BP. This is an *exact* solution approach, i.e., given enough runtime, the optimal solution can be obtained. The models are implemented under CPLEX. The authors report the average runtime. The used benchmark instances are randomly generated and are not part of an available benchmark set, and a comparison is therefore not possible.

(Cid-Garcia and Rios-Solis, 2020) developed the two-stage exact Positions and Covering (P&C) algorithm for the 2BP with and without rotation. The first stage involves an initial placement strategy to assign items to preliminary positions, which maximizes space utilization. The second stage employs a covering algorithm to further optimize item placement and minimize wasted space. Results are reported for the beng and some of the class instances which, necessarily, are the correct optimal solutions (but are only given as averages, sadly). The time limit for the runs was set to 5 hours and the algorithm could not be completed on some instances.

Most recently, (Li et al., 2021) proposed a hybrid adaptive GA (HAGA) for a two-dimensional rectangular packing problem. As in (Bengtsson, 1982), the goal is to maximize the filling rate of the sheets, meaning that some items may not be selected for inclusion. The benchmark instances used are also not in 2DPackLib.

From this brief overview, we immediately notice several problems for any researcher delving into the 2BP. The existing works have different goals (number of bins, fill rate), report results obtained on different benchmarks, use different termination criteria that can either be based on FEs or on time and then, range from 0.5s to many days, and perform different numbers of runs per instance. The actual solutions, the packing plans, are almost never provided. Even worse, the results are always averaged over benchmark instance sets, often normalized using different lower bounds, and usually rounded to one or two decimals. While it is possible (although error-prone) to de-normalize the results for comparison purposes by multiplying with the (right) lower bound, it is not possible to de-average or de-round them…

For a problem as important and common as the 2BP, there should be a complete set of unrounded, unnormalized, and un-averaged results on a standardized and publically available dataset. It is not necessary that such a set represents the state-of-the-art or is continuously updated. The presence of instances together with solutions and objective values alone will allow other researchers to verify and replicate each others' work. And if all future publications include complete results in immutable archives as we do here in https://doi.org/10.5281/zenodo.13324219, the set of best-known solutions emerges automatically.

We chose the recent 2DPackLib benchmark (Iori et al., 2022) for our work, which has been published by researchers who are responsible for several of the most important milestones in the field (Iori et al., 2021; Dell'Amico et al., 2002; Lodi et al., 2002; Lodi et al., 2004; Martello and Vigo, 1998; Monaci and Toth, 2006). Unfortunately, no results or solutions were published along with the benchmark instances. We close this gap. Moreover, by reporting all improving moves of our algorithm for a large computational budget, arbitrarily shorter computational budgets can be simulated by cutting off the later improvements.

4 OUR APPROACH

4.1 RLS and FRLS

As the baseline algorithm for our study, we use the simplest local search method available, Randomized Local Search (RLS), often also called Hill Climbing or $(1+1)$ EA (Russell and Norvig, 2002; Neumann and Wegener, 2007; Johnson et al., 1988). As a black-box metaheuristic, it allows us to choose a search space $\mathbb{P}$ and a search operator $\mathsf{move}: \mathbb{P} \mapsto \mathbb{P}$, a decoding function $\mathsf{decode}: \mathbb{P} \mapsto \mathbb{S}$ that translates the points in the search space to packing plans, and an objective function $f: \mathbb{S} \mapsto \mathbb{N}$ rating the quality of such plans.

The blueprint of this metaheuristic is illustrated in Algorithm 1. The algorithm begins by sampling a random point π_c from the search space $\mathbb{P}$, decoding it to a packing plan s_c, and evaluating its objective value $z_c = f(s_c)$. In a loop, a new point π_n is sampled as a modified copy of π_c using the unary operator move, decoded, and evaluated. If π_n is not worse than π_c, it replaces it. When the computational budget of 10^8 FEs is exhausted, both the best-so-far solution s_c and its quality z_c are returned. In our experiments, the algorithm terminates after 10^8 objective function evaluations (FEs).

FFA is an algorithm module that prescribes replacing the objective values with their encounter frequencies in the selection decisions. Plugging it into the RLS yields the FRLS sketched in Algorithm 2. This algorithm starts like RLS, but additionally initializes a frequency table H to be filled with zeros. Where RLS compares the objective values z_n and z_c to decide whether π_n should replace π_c or be discarded, FRLS first increments the encounter frequencies $H[z_n]$ and $H[z_c]$ of z_n and z_c and then compares these instead of the objective values. As a result, it will accept π_n if it corresponds to a solution whose objective value has been seen less or equally often than the one corresponding to π_c. Since FRLS does not care whether z_n is better than z_c or not, the algorithm may lose the best-discovered solution again and thus needs to remember it in an additional variable s_b.

(Weise et al., 2021b; Weise et al., 2023) discuss the interesting theoretical features of the resulting algorithm that no longer optimizes towards better solutions but, yet, will find these nevertheless because good solutions have rare objective values. It was shown that this scheme yields remarkable perfor-

Algorithm 1: RLS($\mathsf{decode}:\mathbb{P}\mapsto\mathbb{S}, f:\mathbb{S}\mapsto\mathbb{N}$).

sample π_c from $\mathbb{P}$ u.a.r.;
$s_c \leftarrow \mathsf{decode}(\pi_c)$; ▷ *see Algorithm 3*
$z_c \leftarrow f(s_c)$; ▷ *one of the objective functions*
for 10^8-1 times **do** ▷ *our termination criterion*
  $\pi_n \leftarrow \mathsf{move}(\pi_c)$; ▷ *depends on problem type*
  $s_n \leftarrow \mathsf{decode}(\pi_n)$; $z_n \leftarrow f(s_n)$;
  if $z_n \leq z_c$ **then**
    $\pi_c \leftarrow \pi_n$; $s_c \leftarrow s_n$; $z_c \leftarrow z_n$
return s_c, z_c

Algorithm 2: FRLS($\mathsf{decode}:\mathbb{P}\mapsto\mathbb{S}, f:\mathbb{S}\mapsto\mathbb{N}$).

$H \leftarrow (0,0,\cdots,0)$; ▷ *H-table initially all 0s*
sample π_c from $\mathbb{P}$ u.a.r.;
$s_c \leftarrow \mathsf{decode}(\pi_c)$; ▷ *see Algorithm 3*
$z_c \leftarrow f(s_c)$; ▷ *one of the objective functions*
$s_b \leftarrow s_c$; $z_b \leftarrow z_c$; ▷ *best may otherwise get lost*
for 10^8-1 times **do** ▷ *our termination criterion*
  $\pi_n \leftarrow \mathsf{move}(\pi_c)$; ▷ *depends on problem type*
  $s_n \leftarrow \mathsf{decode}(\pi_n)$; $z_n \leftarrow f(s_n)$;
  if $z_n < z_b$ **then** $s_b \leftarrow s_n$; $z_b \leftarrow z_n$;
  $H[z_c] \leftarrow H[z_c]+1$; $H[z_n] \leftarrow H[z_n]+1$;
  if $H[z_n] \leq H[z_c]$ **then**
    $\pi_c \leftarrow \pi_n$; $s_c \leftarrow s_n$; $z_c \leftarrow z_n$
return s_b, z_b ▷ *return preserved best*

mance on the Max-SAT domain, where it can speed up multiple algorithms several thousand times (Weise et al., 2023), as well as on the JSSP (Weise et al., 2021a; de Bruin et al., 2023) and on the TSP (Liang et al., 2022; Liang et al., 2024). Whether it can repeat this impressive performance on the 2BP will be investigated in our experiments.

4.2 Encoding and Search Operators

Defining search operators for the packing plans $s \in \mathbb{S}$ directly is complicated. However, when solving the 2BP|O|F, we can use permutations π with repetitions as search space $\mathbb{P}$ to represent the packing orders. Each item ID $i \in \{1..N\}$ occurs d_i times in π. The permutations π therefore have length T.

For the 2BP|R|F, we allow the elements of π to be signed: Each time an item ID occurs, it then can do so either in its original (positive) value, meaning that an instance of i is to be packed in its original orientation (w_i, h_i), or negated, i.e., as $-i$, which signifies that an instance of i is packed after a $90°$ rotation, that is, having dimensions (h_i, w_i).

The decoding function decode is based on the Improved Bottom Left IBL heuristic by (Liu and Teng, 1999) and adopted to the 2BP in (Zhao et al., 2024). It accepts one such packing order $\pi \in \mathbb{P}$ and translates

Algorithm 3: $\mathsf{decode}(\pi \in \mathbb{P}) \mapsto \mathbb{S}$.

$b \leftarrow 1$; ▷ *start at bin 1*
for $k \in \{1..T\}$ **do** ▷ *iterate over all T elements in π*
  $i \leftarrow \pi[k]$; ▷ *get current item ID*
  if $i < 0$ **then** ▷ *$\equiv$ rotation in 2BP|R|F*
    $i \leftarrow -i$; $w \leftarrow h_i$; $h \leftarrow w_i$;
  else $w \leftarrow w_i$; $h \leftarrow h_i$;
  ▷ *some rotations may be invalid in 2BP|R|F*
  if $w > W \vee h > H$ **then** swap w and h;
  ▷ *put item with bottom-right corner on top-right corner of current bin*
  set $s[k].b \leftarrow b$; $s[k].i \leftarrow i$; $s[k].x_1 \leftarrow W - w$; $s[k].y_1 \leftarrow H$; $s[k].x_2 \leftarrow W$; $s[k].y_2 \leftarrow H + h$;
  move $s[k]$ down-left as far as possible without creating any overlap; prefer moving down over moving left;
  if $s[k].y_2 > H$ **then** ▷ *item does not fit in bin?*
    ▷ *put it at bottom-left corner of a new bin*
    $b \leftarrow b+1$;
    set $s[k].b \leftarrow b$; $s[k].x_1 \leftarrow 0$; $s[k].y_1 \leftarrow 0$; $s[k].x_2 \leftarrow w$; $s[k].y_2 \leftarrow h$;
return s

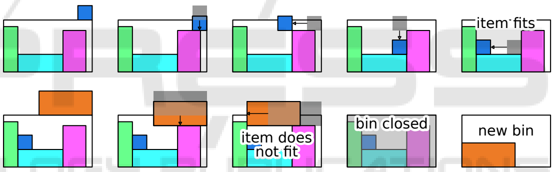

Figure 1: An illustrative example of the decoding Algorithm 3 (read from the top-left to the bottom-right).

it to a packing plan $s \in \mathbb{S}$. It therefore iterates over the (potentially signed) permutation s and places the items into the packing plan in the prescribed order. As sketched in Figure 1, it first places the item instance outside on top of the bin, with its bottom-right corner onto the top-right corner of the current bin b. It then moves the instance downwards and leftwards as far as possible, prioritizing the downward movement whenever possible. If the item instance cannot be moved any further, we check if it is completely contained in the bin b. If yes, it can remain there. Otherwise, a new bin is opened ($b \leftarrow b+1$) and the item instance is placed at its bottom-left corner. Once a new bin is opened, this bin is used for all further insertions.

For the 2BP|O|F, the unary search operator $\mathsf{move}:\mathbb{P}\mapsto\mathbb{P}$ accepts one point $\pi_a \in \mathbb{P}$ and returns a point $\pi_b \in \mathbb{P}$ where two randomly chosen *different* elements are swapped.

For the 2BP|R|F, where the elements of the permutations π can be signed, it produces a point π_b where either one value is negated or two different values are swapped. It therefore first creates a copy π_b of π_a and draws an index j uniformly at random (u.a.r.) in $\{1..T\}$. It then draws a Boolean value v, which is either `True` or `False`, u.a.r. If $v = $ `True`, it tries to swap two different elements in π_b. It therefore attempts for at most $10T$ times to draw a random index $k \in \{1..T\}$ with $\pi_b[k] \neq \pi_b[j]$. If this succeeds, it swaps the values at indices j and k in π_b and returns π_b. Otherwise, i.e., if either no appropriate index k was found (which can happen, e.g., if all items are identical) or if $v = $ `False`, it flips the sign of $\pi_b[j]$ and returns π_b.

Finally, in (Zhao et al., 2024), several objective functions that minimize the number of bins were discussed as alternatives to f_1 on the 2BP|R|F. It was found that the function f_7 (see Equation 5) combining the number of bins f_1 with the area under the skyline (the top border of the packing) in the last bin yielded the best results. An RLS using this objective function will prefer a packing s_n over a packing s_c if it requires fewer bins or, if both require the same number of bins but s_n has a lower skyline in its last bin. We will investigate both f_1 and f_7 in our experiments.

$$\mathrm{inB}(s,b) = \{j\,\forall j \in \{1..T\} \wedge s[j].b = b\} \tag{3}$$

$$\mathrm{sl}(s,b) = \sum_{x=0}^{W-1} \max \left\{ \begin{array}{r} s[j].y_2 : \forall j \in \mathrm{inB}(s,b) \wedge \\ s[j].x_1 \leq x < s[j].x_2 \end{array} \right\} \tag{4}$$

$$f_7(s) = WH(f_1(s) - 1) + \mathrm{sl}(s, f_1(s)) \tag{5}$$

5 EXPERIMENTS AND RESULTS

5.1 Setup

We implement our algorithms in Python 3.10 on Windows 10 on an Intel64 Family 6 Model 167 CPU using the moptipy (Weise and Wu, 2023) framework, as well as numba just-in-time compilation where possible. We conduct 3 runs per algorithm setup and problem instance, except for the RLS-f_7, for which we conduct 5 runs. Since this algorithm performed best, we deemed it worth to gather more data for it as basis for future experiments. We use a computational budget of at most 10^8 objective function evaluations (FEs) per run, which is the same as in (Zhao et al., 2024).

We use the 2DPackLib by (Iori et al., 2022), which offers three sets of 2BP instances in a unified format: The ten instances of type beng (Bengtsson, 1982) have both bin and item dimensions drawn from uniform distributions. They have 20 to 200 items and the largest bin size is $(40,25)$. The class instance set (Berkey and Wang, 1987; Martello and Vigo, 1998) is divided into ten classes based on the bin and item dimensions, the former of which ranges from $(10,10)$ to $(300,300)$. Each class is divided into five groups with $N \in \{20,40,60,80,100\}$ items. Each group contains ten benchmark instances. Finally, the set A (Macedo et al., 2010) offers 43 instances with $N \in \{13..809\}$ and bin sizes of either $(2750,1220)$, $(2550,2100)$, or $(2470,2080)$.

Additionally to the 2DPackLib, we consider the four non-trivial "Almost Squares in Almost Squares" (Asqas$N$) instances from (van den Berg et al., 2016). They have $N \in \{3,8,20,34\}$ and the widths of all objects are one unit larger than their heights, i.e., $w_i = h_i + 1\ \forall i \in \{1..N\}$ and $W = H + 1$. The goal is to pack all items into a single bin, which would result in a perfect packing without any wasted space.

5.2 Results

In Table 1 we present the results of our methods for the 2BP|R|F (with rotation) and in Table 2 for the 2BP|O|F (without rotation).[2] We conducted 3 runs per algorithm setup and problem instance for at most 10^8 objective function evaluations (FEs), except for the RLS-f_7 setups, for which we performed 5 runs. While we do present the sum of the number of bins added up over groups of instances like the related works do, we first average the results over each instance. The final results are rounded to full integers. The best values per instance group are marked with **bold** face and we count how often each algorithm can achieve the best result on the class and beng instance groups in the bottom row (# best c&b).

Among our own four algorithm setups, the RLS using f_7 can achieve the best results. On the 2BP|R|F, it is outperformed only by the BRKGA-2r, which achieves the best result 52 times whereas the average result of RLS-f_7 is best 38 times. The third best algorithm is HHA-NO(sr) (Beyaz et al., 2015), which achieves the best result 22 times.

On the 2BP|O|F, the BRKGA-aNB scores best 49 times, followed by the EA-LGFi by (Blum and Schmid, 2013) (44 times) and the GRASP/VND (Parreño et al., 2010) (43 times). The average results of RLS-f_7 are the best 41 times.

[2] For MXGA (Lee, 2008) normalized results have been reported. We de-normalized them, but had to use lb_m instead of lb_d to get reasonable values. Still, due to rounding errors, we had to correct the values for class 1/60 and class 1/100, which, probably due to the result of rounding in (Lee, 2008), de-normalizing, and then rounding again, came out slightly below the optimal result delivered by P&C. Due to this rounding process, the comparison with MXGA must be taken with a grain of salt. As said, this process is error-prone.

Table 1: Average total number of bins of the RLS and FRLS on the 2BP|R|F (with rotation) for objective functions f_1 and f_7 for 3 runs per setting in comparison to the related work. For RLS-f_7, 5 runs were conducted, for the other settings only 3. Column RLS-$f_7^\star$ is the *best* result of the 5 RLS-f_7 runs.

instance	BRKGA	HHA-NO		MXGA	P&C	FRLS		RLS		
group	$2r$	(r)	(sr)			f_1	f_7	f_1	f_7	$f_7^\star$
a/small						100	99	99	**97**	**97**
a/med						203	221	198	176	**174**
a/large						745	867	721	651	**644**
beng/1-8	**54**				**54**	61	**54**	60	**54**	**54**
beng/9-10	**13**				**13**	15	**13**	14	**13**	**13**
class 1/20	**66**	**66**	**66**	**66**	**66**	**66**	**66**	**66**	**66**	**66**
class 1/40	**128**	131	129	129	**128**	132	129	131	129	**128**
class 1/60	**195**	196	**195**	**195**	**195**	205	**195**	202	**195**	**195**
class 1/80	**270**	**270**	**270**	**270**	**270**	283	274	278	**270**	**270**
class 1/100	**313**	314	**313**	**313**	**313**	340	325	331	**313**	**313**
class 2/20	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
class 2/40	**19**	20	**19**	21	**19**	20	**19**	20	**19**	**19**
class 2/60	**25**	**25**	**25**	**25**	**25**	29	**25**	28	**25**	**25**
class 2/80	**31**	**31**	**31**	**31**	**31**	36	**31**	35	**31**	**31**
class 2/100	**39**	**39**	**39**	**39**	**39**	42	40	42	**39**	**39**
class 3/20	**47**	48	48	48	**47**	48	**47**	**47**	**47**	**47**
class 3/40	**92**	95	95	94		97	94	97	94	93
class 3/60	**134**	137	137	136		148	142	146	**134**	**134**
class 3/80	**182**	186	187	184		207	206	201	183	183
class 3/100	**220**	225	225	223		250	253	244	**220**	**220**
class 4/20	**10**	**10**	**10**	**10**		**10**	**10**	**10**	**10**	**10**
class 4/40	**19**	**19**	**19**	**19**		**19**	**19**	**19**	**19**	**19**
class 4/60	**23**	25	25	25		28	25	27	24	**23**
class 4/80	**31**	32	33	32		35	33	35	**31**	**31**
class 4/100	**37**	38	38	38		42	40	42	**37**	**37**
class 5/20	**59**	**59**	**59**	**59**		**59**	**59**	**59**	**59**	**59**
class 5/40	**114**	116	115	**114**		120	118	119	**114**	**114**
class 5/60	**172**	175	176	177		186	191	182	174	173
class 5/80	**239**	240	241	241		257	271	251	**239**	**239**
class 5/100	**277**	284	284	279		308	330	302	278	278
class 6/20	**10**	**10**	**10**	**10**		**10**	**10**	**10**	**10**	**10**
class 6/40	**16**	18	17	21		19	17	19	**16**	**16**
class 6/60	**21**	22	22	**21**		23	22	23	**21**	**21**
class 6/80	**30**	**30**	**30**	**30**		31	**30**	31	**30**	**30**
class 6/100	**32**	34	34	34		39	37	38	**32**	**32**
class 7/20	**52**	**52**	**52**	**52**		**52**	**52**	**52**	**52**	**52**
class 7/40	**102**	106	107	104		111	109	107	103	**102**
class 7/60	**146**	152	153	147		162	164	157	**146**	**146**
class 7/80	**208**	216	217	213		232	244	228	**208**	**208**
class 7/100	**250**	260	259	255		281	298	274	**250**	**250**
class 8/20	**53**	**53**	**53**	**53**		**53**	**53**	**53**	**53**	**53**
class 8/40	**103**	106	105	105		111	111	108	104	104
class 8/60	**147**	155	154	149		164	166	160	148	148
class 8/80	**204**	213	214	209		229	240	225	207	206
class 8/100	**252**	261	262	255		284	302	278	253	**252**
class 9/20	**143**	**143**	**143**	**143**		**143**	**143**	**143**	**143**	**143**
class 9/40	**275**	**275**	**275**	**275**		**275**	**275**	**275**	**275**	**275**
class 9/60	**435**	**435**	**435**	436		**435**	**435**	**435**	**435**	**435**
class 9/80	**573**	**573**	**573**	574		**573**	**573**	**573**	**573**	**573**
class 9/100	**693**	**693**	**693**	695		**693**	**693**	**693**	**693**	**693**
class 10/20	**41**	**41**	**41**	43		**41**	**41**	**41**	**41**	**41**
class 10/40	**72**	73	73	73		79	73	76	73	**72**
class 10/60	**99**	101	101	101		112	107	109	**99**	**99**
class 10/80	**125**	129	130	129		146	144	142	126	**125**
class 10/100	**154**	161	162	159		184	185	179	158	156
asqas						**6**	**6**	**6**	**6**	**6**
# best c&b	52	19	22	19	13	14	22	15	38	44

Table 2: Average total number of bins of the RLS and FRLS on the 2BP|O|F (without rotation) for objective functions f_1 and f_7 in comparison to the related work. For RLS-f_7, 5 runs were conducted, for the other settings only 3. Column RLS-$f_7^\star$ is the *best* result of the 5 RLS-f_7 runs.

instance	BRKGA	EA	GRASP	P&C	FRLS		RLS		
group	*aNB*	LGFi	VND		f_1	f_7	f_1	f_7	$f_7^\star$
a/small					102	**101**	**101**	**101**	**101**
a/med					202	211	202	181	**180**
a/large					733	801	719	641	**635**
beng/1-8	**54**		**54**	**54**	60	**54**	60	55	**54**
beng/9-10	**13**		**13**	**13**	14	**13**	14	**13**	**13**
class 1/20	**71**	**71**	**71**	**71**	**71**	**71**	**71**	**71**	**71**
class 1/40	**134**	**134**	**134**	**134**	136	**134**	136	**134**	**134**
class 1/60	**200**	**200**	**200**	**200**	206	**200**	203	**200**	**200**
class 1/80	**275**	**275**	**275**	**275**	285	**275**	281	**275**	**275**
class 1/100	**317**	**317**	**317**	**317**	341	324	333	**317**	**317**
class 2/20	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
class 2/40	**19**	**19**	**19**	**19**	20	**19**	20	**19**	**19**
class 2/60	**25**	**25**	**25**	**25**	28	**25**	27	**25**	**25**
class 2/80	**31**	**31**	**31**	**31**	36	**31**	35	**31**	**31**
class 2/100	**39**	**39**	**39**	**39**	42	**39**	42	**39**	**39**
class 3/20	**51**	**51**	**51**	**51**	**51**	**51**	**51**	**51**	**51**
class 3/40	**94**	**94**	**94**		98	**94**	97	**94**	**94**
class 3/60	**139**	**139**	**139**		149	140	147	140	**139**
class 3/80	**189**	**189**	**189**		206	204	204	190	**189**
class 3/100	**223**	224	**223**		251	249	243	**223**	**223**
class 4/20	**10**	**10**	**10**		**10**	**10**	**10**	**10**	**10**
class 4/40	**19**	**19**	**19**		**19**	**19**	**19**	**19**	**19**
class 4/60	25	**23**	25		28	25	27	25	24
class 4/80	**31**	**31**	**31**		36	33	36	**31**	**31**
class 4/100	**37**	**37**	38		42	39	42	**37**	**37**
class 5/20	**65**	**65**	**65**		**65**	**65**	**65**	66	**65**
class 5/40	**119**	**119**	**119**		123	120	122	**119**	**119**
class 5/60	**180**	**180**	**180**		188	190	185	**180**	**180**
class 5/80	**247**	**247**	**247**		259	270	253	**247**	**247**
class 5/100	**281**	284	282		309	328	305	282	**281**
class 6/20	**10**	**10**	**10**		**10**	**10**	**10**	**10**	**10**
class 6/40	**16**	17	17		19	17	19	17	**16**
class 6/60	**21**	**21**	**21**		23	22	23	**21**	**21**
class 6/80	**30**	**30**	**30**		32	**30**	31	**30**	**30**
class 6/100	33	**32**	34		39	37	38	**32**	**32**
class 7/20	**55**	**55**	**55**		**55**	**55**	**55**	**55**	**55**
class 7/40	**111**	**111**	**111**		116	113	116	**111**	**111**
class 7/60	**158**	159	159		165	162	162	159	**158**
class 7/80	**232**	**232**	**232**		239	240	236	**232**	**232**
class 7/100	**271**	**271**	**271**		284	293	282	**271**	**271**
class 8/20	**58**	**58**	**58**		**58**	**58**	**58**	**58**	**58**
class 8/40	**113**	**113**	**113**		115	114	115	**113**	**113**
class 8/60	**161**	**161**	**161**		168	167	166	**161**	**161**
class 8/80	**224**	**224**	**224**		233	236	231	**224**	**224**
class 8/100	278	**277**	278		292	298	286	**277**	**277**
class 9/20	**143**	**143**	**143**		**143**	**143**	**143**	**143**	**143**
class 9/40	**278**	**278**	**278**		**278**	**278**	**278**	**278**	**278**
class 9/60	**437**	**437**	**437**		**437**	**437**	**437**	**437**	**437**
class 9/80	**577**	**577**	**577**		**577**	**577**	**577**	**577**	**577**
class 9/100	**695**	**695**	**695**		**695**	**695**	**695**	**695**	**695**
class 10/20	**42**	**42**	**42**		43	**42**	**42**	**42**	**42**
class 10/40	**74**	**74**	**74**		79	**74**	77	**74**	**74**
class 10/60	**100**	101	**100**		112	105	111	101	**100**
class 10/80	**128**	**128**	129		145	143	142	129	**128**
class 10/100	**158**	160	159		183	184	179	159	159
asqas					**8**	**8**	**8**	**8**	**8**
# best c&b	49	44	43	13	14	27	15	41	50

However, if we consider the best results of five runs of RLS-f_7 (denoted in column $f_7^\star$), this algorithm achieves 50 times the best solution and would rank first. In other words, had we given five times the computational budget and performed restarts, a simple local search would have outperformed all of the much more complicated algorithm designs on the 2BP|O|F.

The P&C (Cid-Garcia and Rios-Solis, 2020) is an exact method that always finds the optimal solutions. On the 2BP|R|F, the *average* results of RLS-f_7 reach the same (optimal) quality on all but one of the instance groups (class 1/40) where results of P&C are available. The best result of the RLS with f_7 (column $f_7^\star$) is also optimal on class 1/40. The exact same situation can be observed on the 2BP|O|F, but now beng/1-8 is the only instance group for which P&C results are available where RLS-f_7 is worse (on average). The best of five runs of the same algorithm do find the optimal solutions for beng/1-8 as well.

It becomes obvious from both tables that f_7 leads to much better results compared to f_1. This is expected. What is unexpected is that FRLS is consistently worse than RLS. RLS is prone to get stuck at local optima. From the tables, we also know that the

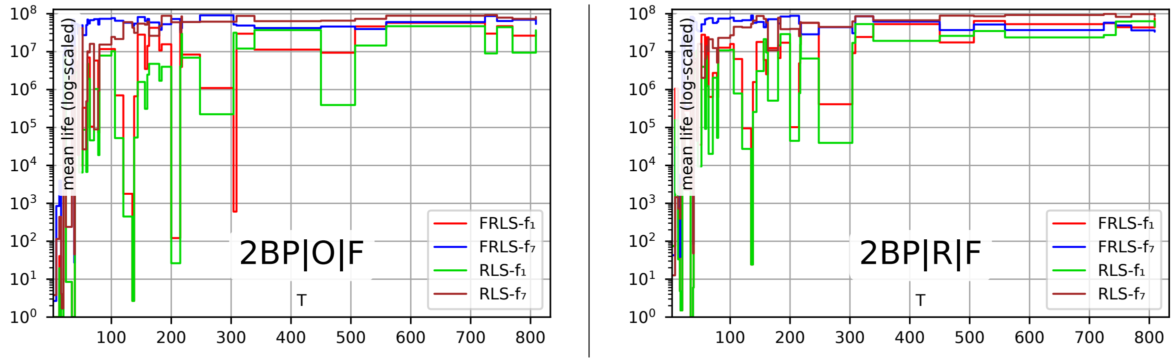

Figure 2: The average over the index *life* of the objective function evaluation where the last improving move was made by the different algorithms, plotted over the total number T of item instances to pack. The plot for the problem variant *without* rotation (2BP|O|F) is on the left and the one for the variant *with* rotation (2BP|R|F) is on the right.

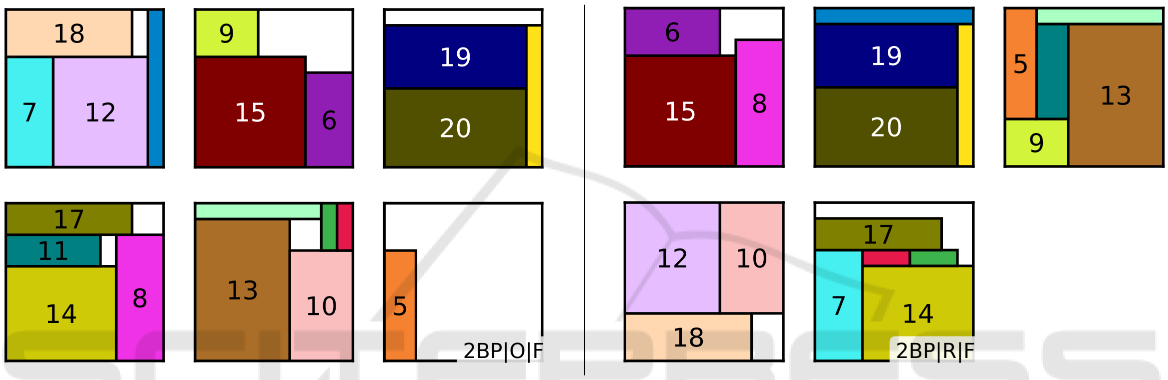

Figure 3: Two optimal result of RLS-f_7 on class 1/20-4 for the problem variant *without* rotation (2BP|O|F) on the left and for the variant *with* rotation (2BP|R|F, on the right).

best-of-five-runs result ($f_7^\star$ columns) is better than the average results (f_7 columns) for RLS. One would assume that this must be due to RLS getting stuck at different local optima. (In this case, restarting the algorithm, which is equivalent to running it five times and taking the best result, would be a good idea.) Then why does FRLS, which has been shown to deliver much better results in previous works on $\mathcal{NP}$-hard problems, perform worse than RLS? Should it not be able to escape from the local optima?

Now, in our experiments, we use a generous budget of 10^8 objective function evaluations per run. If RLS would get stuck at local optima, then we would expect that it would stop improving much earlier. The index *life* of the objective function evaluation where its last improving move takes place would be much lower than 10^8. Of course, it will also be lower on small-scale instances where it already finds the optimal solution and we know that it does so on several instances from our tables. Either way, in Figure 2 we plot this index over the total number T of item instances to pack. In stark contrast to the reasonable expectation regarding the behavior of RLS-f_7, we notice that the algorithm keeps finding improvements until the end of the computational budget, consistently over all not-too-small problem scales and for both investigated objective functions. RLS-f_7 does *not* get stuck in local optima. As a result, FRLS cannot outperform it, because its strength is exactly to avoid getting stuck at local optima and it trades in speed for obtaining this ability.

Finally, we illustrate two packings discovered by RLS-f_7 on instance class 1/20-4. From Table 1 and Table 2, we know that RLS-f_7 finds results of the same average quality over all class 1/20 instances as P&C. Since P&C is an exact method always returning the optimal solution, the results of RLS-f_7 are therefore also optimal in average and, hence, optimal in each run on each of the class 1/20 instances. The optimal packing in Figure 3 for the problem variant 2BP|O|F where items have a fixed orientation and cannot be rotated and requires six bins. The fact that lots of space in the bins is left unused hints towards the problem being rather easy, as this would probably allow us to place the items slightly differently and still get the optimal number of bins. Figure 3 also shows the optimal packing for the same instance but for 2BP|R|F, illustrating that one bin can be saved if item rotation is permitted.

6 CONCLUSIONS

Our experiments confirmed that randomized local search (RLS) performs very well on two important variants of the 2BP, the 2BP|R|F and the 2BP|O|F. What does this mean? Our results indicate that RLS does not get stuck at local optima, or, at least does so much later than one would expect. The implications of this are interesting: If RLS does not get suck at local optima, then *any mechanism that aims to avoid getting stuck at local optima is essentially useless*. If RLS does not get stuck at local optima, then extending it to tabu search by introducing tabu and aspiration criteria to avoid local optima could not yield a performance improvement. If RLS does not get stuck at local optima, then sometimes accepting worse solutions, as simulated annealing would do, could not yield better results. If RLS does not get stuck at local optima, then FRLS cannot outperform it. The latter is what we observed in our experiments here as well, while doing tests with tabu search and simulated annealing will be part of our future work. Also, this could be the reason why the very simple RLS seems to be competitive to much more sophisticated algorithms.

Now it seems unlikely that an $\mathcal{NP}$-hard problem does not have local optima. But maybe the benchmark instances in the 2DPackLib are not challenging enough. So from this perspective, we would suggest that creating harder instances is indeed needed. This, too, is part of our future work.

Therefore, while this work presents a first negative result for Frequency Fitness Assignment (FFA), this finding has to be taken with a grain of salt: We will revisit the problem once we have a set of instances of which we can confirm that RLS cannot solve them well.

ACKNOWLEDGEMENTS

The authors acknowledge support from the Project of National Natural Science Foundation of China 62406095, the Project of Natural Science Foundation of Anhui Province 2308085MF213, the Key Research Plan of Anhui Province 2022k07020011, the University Scientific Research Innovation Team Project of Anhui Province 2022AH010095, as well as the Hefei Specially Recruited Foreign Expert program and the Hefei Foreign Expert Office program.

REFERENCES

Bengtsson, B.-E. (1982). Packing rectangular pieces – A heuristic approach. *The Computer Journal*, 25(3):353–357.

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

Beyaz, M., Dokeroglu, T., and Cosar, A. (2015). Robust hyper-heuristic algorithms for the offline oriented/non-oriented 2D bin packing problems. *Applied Soft Computing*, 36:236–245. doi:10.1016/j.asoc.2015.06.063.

Blum, C. and Schmid, V. (2013). Solving the 2D bin packing problem by means of a hybrid evolutionary algorithm. In Alexandrov, V., Lees, M., Krzhizhanovskaya, V. V., Dongarra, J. J., and Sloot, P. M. A., editors, *International Conference on Computational Science (ICCS'13), June 5–7, 2013, Barcelona, Spain*, volume 18 of *Procedia Computer Science*, pages 899–908, Amsterdam, The Netherlands. Elsevier. doi:10.1016/j.procs.2013.05.255.

Boschetti, M. A. and Mingozzi, A. (2003). The two-dimensional finite bin packing problem. Part II: New lower and upper bounds. *4OR – the Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(2):135–147. doi:10.1007/s10288-002-0006-y.

Braam, F. and van den Berg, D. (2022). Which rectangle sets have perfect packings? *Operations Research Perspectives*, 9(100211). doi:10.1016/j.orp.2021.100211.

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

de Bruin, E., Thomson, S. L., and van den Berg, D. (2023). Frequency fitness assignment on JSSP: A critical review. In Correia, J., Smith, S. L., and Qaddoura, R., editors, *Proceedings of the 26th European Conference on Applications of Evolutionary Computation (EvoApplications'23), Held as Part of EvoStar 2023, April 12-14, 2023, Brno, Czech Republic*, volume 13989 of *Lecture Notes in Computer Science*, pages 351–363, Cham, Switzerland. Springer. doi:10.1007/978-3-031-30229-9_23.

Dell'Amico, M., Martello, S., and Vigo, D. (2002). A lower bound for the non-oriented two-dimensional bin packing problem. *Discrete Applied Mathematics*, 118(1-2):13–24. doi:10.1016/S0166-218X(01)00253-0.

Gonçalves, J. F. and Resende, M. G. (2013). A biased random key genetic algorithm for 2D and 3D bin packing problems. *International Journal on Production Economics*, 145(2):500–510. doi:10.1016/j.ijpe.2013.04.019.

Iori, M., de Lima, V. L., Martello, S., Miyazaw, F. K., and Monaci, M. (2021). Exact solution techniques for two-dimensional cutting and packing. *European Journal of Operational Research*, 289(2):399–415. doi:10.1016/j.ejor.2020.06.050.

Iori, M., de Lima, V. L., Martello, S., and Monaci, M. (2022). 2DPackLib: A two-dimensional cutting and packing library. *Optimization Letters*, 16(2):471–480. doi:10.1007/s11590-021-01808-y.

Johnson, D. S., Papadimitriou, C. H., and Yannakakis, M. (1988). How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100. doi:10.1016/0022-0000(88)90046-3.

Kierkosz, I. and Luczak, M. (2013). A hybrid evolutionary algorithm for the two-dimensional packing problem. *Central European Journal of Operations Research*, 22(4):729–753.

Lee, L.-S. (2008). A genetic algorithm for two-dimensional bin packing problem. *MathDigest: Research Bulletin of Institute for Mathematical Research*, 2(1):34–39. http://psasir.upm.edu.my/id/eprint/12464/1/Artikel_6_vol2_no1.pdf.

Li, X. and Zhang, K. (2018). Single batch processing machine scheduling with two-dimensional bin packing constraints. *International Journal of Production Economics*, 196:113–121. doi:10.1016/j.ijpe.2017.11.015.

Li, Y., Sang, H., Xiong, X., and Li, Y. (2021). An improved adaptive genetic algorithm for two-dimensional rectangular packing problem. *Applied Sciences*, 11(1):413. doi:10.3390/app11010413.

Liang, T., Wu, Z., Lässig, J., van den Berg, D., Thomson, S. L., and Weise, T. (2024). Addressing the traveling salesperson problem with frequency fitness assignment and hybrid algorithms. *Soft Computing*. doi:10.1007/s00500-024-09718-8.

Liang, T., Wu, Z., Lässig, J., van den Berg, D., and Weise, T. (2022). Solving the traveling salesperson problem using frequency fitness assignment. In Ishibuchi, H., Kwoh, C., Tan, A., Srinivasan, D., Miao, C., Trivedi, A., and Crockett, K. A., editors, *IEEE Symposium Series on Computational Intelligence (SSCI'22), December 4–7, 2022, Singapore*, pages 360–367, Piscataway, NJ, USA. IEEE. doi:10.1109/SSCI51031.2022.10022296.

Liu, D. and Teng, H. (1999). An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. *European Journal of Operational Research*, 112(2):413–420. doi:10.1016/S0377-2217(97)00437-2.

Lodi, A., Martello, S., and Vigo, D. (2002). Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, 123(1–3):379–396. doi:10.1016/S0166-218X(01)00347-X.

Lodi, A., Martello, S., and Vigo, D. (2004). TSpack: A unified tabu search code for multi-dimensional bin packing problems. *Annals of Operations Research*, 131(1–4):203–213. doi:10.1023/B:ANOR.0000039519.03572.08.

Ma, N. and Zhou, Z. (2017). Mixed-integer programming model for two-dimensional non-guillotine bin packing problem with free rotation. In *4th International Conference on Information Science and Control Engineering (ICISCE), July 21-23, 2017, Changsha, China*, pages 456–460, Piscataway, NJ, USA. IEEE. doi:10.1109/ICISCE.2017.102.

Macedo, R., Alves, C., and Valério de Carvalho, J. M. (2010). Arc-flow model for the two-dimensional guillotine cutting stock problem. *Computers & Operations Research*, 37(6):991–1001. doi:10.1016/j.cor.2009.08.005.

Martello, S. and Vigo, D. (1998). Exact solution of the two-dimensional finite bin packing problem. *Management Science*, 44(3):388–399. doi:10.1287/mnsc.44.3.388.

Monaci, M. and Toth, P. (2006). A set-covering-based heuristic approach for bin-packing problems. *INFORMS Journal on Computing*, 18(1):1–134. doi:10.1287/ijoc.1040.0089.

Neumann, F. and Wegener, I. (2007). Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science*, 378(1):32–40. doi:10.1016/j.tcs.2006.11.002.

Parreño, F., Alvarez-Valdés, R., Oliveira, J. F., and Tamarit, J. M. (2010). A hybrid GRASP/VND algorithm for two- and three-dimensional bin packing. *Annals of Operations Research*, 179(1):203–220. doi:10.1007/s10479-008-0449-4.

Pejic, I. and van den Berg, D. (2020). Monte carlo tree search on perfect rectangle packing problem instances. In Coello, C. A. C., editor, *Genetic and Evolutionary Computation Conference (GECCO'20), Companion Volume, July July 8-12, 2020, Cancún, Mexico*, pages 1697–1703, New York, NY, USA. ACM. doi:10.1145/3377929.3398115.

Pinto, M., Silva, C., Thürer, M., and Moniz, S. (2024). Survey in operations research and management science. nesting and scheduling optimization of additive manufacturing systems: Mapping the territory. *Computers & Operations Research*, 165(106592). doi:10.1016/j.cor.2024.106592.

Russell, S. J. and Norvig, P. (2002). *Artificial Intelligence: A Modern Approach (AIMA)*. Prentice Hall International Inc., Upper Saddle River, NJ, USA, 2 edition.

Terashima-Marín, H., Zárate, C. J. F., Ross, P., and Valenzuela-Rendón, M. (2007). Comparing two models to generate hyper-heuristics for the 2D-regular bin-packing problem. In Lipson, H., editor, *Genetic and Evolutionary Computation Conference (GECCO'07), July 7-11, 2007, London, UK*, pages 2182–2189, New York, NY, USA. ACM. doi:10.1145/1276958.1277377.

van den Berg, D., Braam, F., Moes, M., Suilen, E., and Bhulai, S. (2016). Almost squares in almost squares: Solving the final instance. In Bhulai, S. and Semanjski, I., editors, *DATA ANALYTICS 2016: The Fifth International Conference on Data Analytics, October 9-13, 2026, Venice, Italy*, Wilmington, DE, USA. International Academy, Research, and Industry Association (IARIA). https://math.vu.nl/~sbhulai/publications/data_analytics2016b.pdf.

Weise, T., Chiong, R., Tang, K., Lässig, J., Tsutsui, S., Chen, W., Michalewicz, Z., and Yao, X. (2014a).

Benchmarking optimization algorithms: An open source framework for the traveling salesman problem. *IEEE Computational Intelligence Magazine*, 9(3):40–52. doi:10.1109/MCI.2014.2326101.

Weise, T., Li, X., Chen, Y., and Wu, Z. (2021a). Solving job shop scheduling problems without using a bias for good solutions. In *Genetic and Evolutionary Computation Conference (GECCO'21), July 10-14, 2021, Lille, France, Companion Volume*, pages 1459–1466, New York, NY, USA. ACM. doi:10.1145/3449726.3463124.

Weise, T., Wan, M., Tang, K., Wang, P., Devert, A., and Yao, X. (2014b). Frequency fitness assignment. *IEEE Transactions on Evolutionary Computation*, 18(2):226–243. doi:10.1109/TEVC.2013.2251885.

Weise, T. and Wu, Z. (2023). Replicable self-documenting experiments with arbitrary search spaces and algorithms. In *Genetic and Evolutionary Computation Conference Companion (GECCO'23 Companion), July 15-19, 2023, Lisbon, Portugal*, New York, NY, USA. ACM. doi:10.1145/3583133.3596306.

Weise, T., Wu, Z., Li, X., and Chen, Y. (2021b). Frequency fitness assignment: Making optimization algorithms invariant under bijective transformations of the objective function value. *IEEE Transactions on Evolutionary Computation*, 25(2):307–319. doi:10.1109/TEVC.2020.3032090.

Weise, T., Wu, Z., Li, X., Chen, Y., and Lässig, J. (2023). Frequency fitness assignment: Optimization without bias for good solutions can be efficient. *IEEE Transactions on Evolutionary Computation*, 27(4):980–992. doi:10.1109/TEVC.2022.3191698.

Wong, L. and Lee, L. S. (2009). Heuristic placement routines for two-dimensional bin packing problem. *Journal of Mathematics and Statistics*, 5(4):334–341. https://www.thescipub.com/pdf/jmssp.2009.334.341.

Zhao, R., Liang, T., Wu, Z., van den Berg, D., Thürer, M., and Weise, T. (2024). Randomized local search on the 2D rectangular bin packing problem with item rotation. In *Genetic and Evolutionary Computation Conference (GECCO'24 Companion), July 14–18, 2024, Melbourne, VIC, Australia*, pages 235–238, New York, NY, USA. ACM. doi:10.1145/3638530.3654139.