Discrete Optimization

A hybrid feasibility constraints-guided search to the two-dimensional bin packing problem with due dates

Sergey Polyakovskiy[a], Rym M'Hallah[b],*

[a] School of Information Technology, Deakin University, Australia
[b] Department of Statistics and Operations Research, College of Science, Kuwait University, P.O. Box 5969, Safat 13060, Kuwait

ARTICLE INFO

ABSTRACT

The two-dimensional non-oriented bin packing problem with due dates packs a set of rectangular items, which may be rotated by $90°$, into identical rectangular bins. The bins have equal processing times. An item's lateness is the difference between its due date and the completion time of its bin. The problem packs all items without overlap as to minimize maximum lateness L_{max}.

The paper proposes a tight lower bound that enhances an existing bound on L_{max} by 31.30% for 24.07% of the benchmark instances and matches it in 30.87% cases. Moreover, it models the problem via mixed integer programming (MIP), and solves small-sized instances exactly using *CPLEX*. It approximately solves larger-sized instances using a two-stage heuristic. The first stage constructs an initial solution via a first-fit heuristic that applies an iterative *constraint programming* (CP)-based neighborhood search. The second stage, which is iterative too, approximately solves a series of assignment low-level MIPs that are guided by *feasibility constraints*. It then enhances the solution via a high-level random local search. The approximate approach improves existing upper bounds by 27.45% on average, and obtains the optimum for 33.93% of the instances. Overall, the exact and approximate approaches find the optimum in 39.07% cases.

The proposed approach is applicable to complex problems. It applies CP and MIP sequentially, while exploring their advantages, and hybridizes heuristic search with MIP. It embeds a new lookahead strategy that guards against infeasible search directions and constrains the search to improving directions only; thus, differs from traditional lookahead beam searches.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

Bin packing (BP) is a classical strongly $\mathcal{NP}$-hard combinatorial optimization problem (Jansen & Pradel, 2016; Johnson, Demers, Ullman, Garey & Graham, 1974). It consists in packing a set of items into as few bins as possible. Because of its prevalence in industry, BP has engendered many variants. Some variants impose additional constraints on the packing of the items or on the types of bins such as the oriented, orthogonal, guillotine, and variable-sized BP. More recent variants combine BP with further complicating combinatorial aspects. For example, BP appears in combination with routing problems: minimizing transportation costs subject to loading constraints (Iori & Martello, 2013). It also emerges in lock scheduling (Verstichel, Kinable, De Causmaecker & Vanden Berghe, 2015) where lockages are scheduled, chambers are assigned to ships, and ships are positioned into chambers.

Following this trend, this paper addresses the non-oriented two-dimensional BP problem where items have due dates. This problem, denoted hereafter 2BPP with DD, searches for a feasible packing of a given set of n rectangular items into a set of at most $b \leq n$ identical rectangular bins, and schedules their packing as to minimize the maximum lateness L_{max} of the items. Each item is characterized by its width, height, and due date. Its lateness is the difference between its completion time and its due date, where its completion time is that of its assigned bin. All bins' processing times are equal regardless of their assigned items. This problem is common in make-to-order low-volume production systems such as the high-fashion apparel industry and food delivery. In these contexts, packing efficiency might be increased by mixing up several orders; however, the increased efficiency cannot be at the cost of customer service. That is, a company should choose, from the pool of items emanating from all orders, the ones that need to be packed (or cut) simultaneously with the objective of maximizing material utilization (or packing efficiency) while meeting due dates.

* Corresponding author.
E-mail addresses: sergey.polyakovskiy@deakin.edu.au (S. Polyakovskiy), rym.mhallah@ku.edu.kw (R. M'Hallah).

Similar problems were considered in the literature. Reinertsen and Vossen (2010) investigate the one-dimensional cutting stock problem within steel manufacturing where orders have due dates that must be met. Arbib and Marinelli (2017) study a one-dimensional bin packing problem with the objective of minimizing a weighted sum of maximum lateness and maximum completion time. Li (1996) tackles a two-dimensional cutting stock problem where meeting the orders' due dates is more important than minimizing the wasted material. Arbib and Marinelli (2014) survey the state of the art on packing with due dates. Polyakovskiy and M'Hallah (2011) address the on-line cutting of small rectangular items out of large rectangular stock material using parallel machines in a just-in-time environment. The cutting pattern minimizes both material waste and the sum of earliness-tardiness of the items. Polyakovskiy, Makarowsky and M'Hallah (2017) consider another variant of BP. Items that are cut from the same bin form a batch, whose processing time depends on its assigned items. The items of a batch share the completion time of their bin. The problem searches for the cutting plan that minimizes the weighted sum of the earliness and tardiness of the items.

Bennell, Lee and Potts (2013) deal with a bi-criteria version of 2BPP with DD. They minimize simultaneously the number of used bins and the maximum lateness of the items. They propose a lower bound LB_1 to L_{max} and approximately solve their bi-criteria problem using a single-crossover genetic algorithm, a multi-crossover genetic algorithm (*MXGA*), a unified tabu search, and a randomized descent. They generate a benchmark set for which they report the best average value for each of their two objective functions. They conclude that *MXGA* yields consistently the best upper bound L_{max}^{MXGA} on L_{max}.

This paper focuses on minimizing L_{max} only (in lieu of both the number of used bins and L_{max} as the bi-criteria case does). It is in no way a shortcoming for four reasons. First, the number of bins is naturally bounded. Second, for a given feasible bound on the number of used bins b, searching for the minimal L_{max} is a standard practice to tackle multi-objective problems. Thus, it can be applied iteratively to build the Pareto optimal frontier of the bi-criteria problem. Third, the objectives of minimizing lateness and maximizing packing efficiency do not necessarily conflict (Bennell et al., 2013). Finally, it can be used by decision makers as a decision support tool that quantifies the tradeoff between service quality loss and reduction of both ecological cost and waste material.

As for all difficult combinatorial optimization problems, finding an exact solution, in a reasonable time, for large-sized instances of 2BPP with DD is challenging. Indeed, BP variants are generally tackled using approximate approaches that are based on metaheuristics (Lodi, Martello, Monaci & Vigo, 2014; Lodi, Martello & Vigo, 2002; Sim & Hart, 2013), including genetic algorithms, and hyper-heuristics (Burke, Hyde & Kendall, 2006; López-Camacho, Terashima-Marin, Ross & Ochoa, 2014; Sim, Hart & Paechter, 2015). Unlike the aforementioned techniques, the proposed two-stage approximate approach for 2BPP with DD explores the complementary strengths of constraint programming (CP) and mixed integer programming (MIP). In its first stage, it applies CP. In its second stage, it hybridizes heuristic search with MIP, where MIP is in turn guided by feasibility constraints. In addition, it applies an innovative lookahead strategy that (i) forbids searching in directions that will eventually lead to infeasible solutions and (ii) directs the search towards improving solutions only. Consequently, the proposed assignment based packing approach with its new lookahead strategy is a viable alternative to the constructive heuristics traditionally applied to BP, where bins are filled sequentially in a very greedy manner (Lodi et al., 2002).

Section 2 gives a mathematical formulation of the 2BPP with DD. Section 3 provides essential background information on feasibility constraints and on a CP-based approach for the two-dimensional orthogonal packing problem. Section 4 presents the existing lower bounds LB_1 and LB_2 and the new one LB_3. Section 5 proposes the two-stage solution approach with Section 5.1 detailing the first-fit heuristic (i.e., stage one), Section 5.2 describing the assignment based heuristic, and Section 5.3 summarizing the second stage. Section 6 discusses the results of the computational investigation performed on benchmark instances. Finally, Section 7 summarizes the paper and gives some concluding remarks.

2. Mathematical formulation

Let $B = \{1, \ldots, b\}$ be a set of b identical rectangular bins. Bin $k \in B$ has a width W, a height H, and a processing time P. Items assigned to the same bin have a common completion time. Let $N = \{1, \ldots, n\}$ be a set of n rectangular items, where $n \geq b$. Item $i \in N$ has a width $w_i \leq W$, a height $h_i \leq H$, and a due date d_i. When $w_i \leq H$ and $h_i \leq W$, item $i \in N$ may be rotated by 90° for packing purposes. Every item $i \in N$ must be packed without overlap and must be completely contained within its assigned bin. When assigned to bin k, item $i \in N$ has a completion time $C_i = kP$ and lateness $L_i = C_i - d_i$. The 2BPP with DD consists in finding a feasible packing of the n items into the available bins with the objective of minimizing L_{max}, defined by $L_{max} = \max_{i \in N} \{C_i - d_i\}$.

Let N^* denote the set N appended by the rotated duplicates. The duplicate of item $i, i \in N$, is item $n + i$ of width h_i, height w_i and due date d_i. The problem is then modeled as an MIP with six types of variables.

- $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the position of an item within its assigned bin, where $x_i \geq 0$ and $y_i \geq 0, i \in N^*$, are the bottom left coordinates of item i.
- $\boldsymbol{f}$ signals the assignment of an item to a bin, where $f_{ik} = 1$ if item i is packed into bin $k, i \in N^*, k \in B$, and 0 otherwise.
- $\boldsymbol{l}$ and $\boldsymbol{u}$ are binary. They refer to the relative position of two items. $l_{ij} = 1$ (resp. $u_{ij} = 1$), $i \in N^*$, $j \in N^*$, $i \neq j$, $j \neq i + n$, and $i \neq j + n$, is used to make i to the left of (resp. below) j when i and j are in the same bin.
- The sixth is the objective value, which is L_{max}.

When the rotated duplicate of item i cannot fit into a bin, i.e. its $w_i \leq H$ and $h_i \leq W, i \in N$, its corresponding decision variables are not defined; thus, they are omitted from the model; so are any corresponding constraints.

The MIP model (*EXACT*), which uses the disjunctive constraint technique of Chen, Lee and Shen (1995) and Onodera, Taniguchi and Tamaru (1991), follows.

$$\min L_{max} \tag{1}$$

s.t.
$$l_{ij} + l_{ji} + u_{ij} + u_{ji} - f_{ik} - f_{jk} \geq -1,$$
$$(i, j) \in N^{*2}, \ i < j, \ j \neq n + i, \ k \in B \tag{2}$$

$$x_i + w_i \leq x_j + W\left(1 - l_{ij}\right),$$
$$(i, j) \in N^{*2}, i \neq j, \ j \neq i + n, \ i \neq j + n \tag{3}$$

$$y_i + h_i \leq y_j + H\left(1 - u_{ij}\right),$$
$$(i, j) \in N^{*2}, i \neq j, \ j \neq i + n, \ i \neq j + n \tag{4}$$

$$x_i \leq W - w_i, \quad i \in N^* \tag{5}$$

$$y_i \leq H - h_i, \quad i \in N^* \tag{6}$$

$$\sum_{k \in B} (f_{ik} + f_{n+ik}) = 1, \quad i \in N \tag{7}$$

$$\sum_{k \in B} (kP - d_i) f_{ik} \leq L_{max}, \quad i \in N^* \tag{8}$$

$$l_{ij} \in \{0, 1\}, \; b_{ij} \in \{0, 1\},$$
$$(i, j) \in N^{*2}, \; i \neq j, \; j \neq i + n, \; i \neq j + n \tag{9}$$

$$f_{ik} \in \{0, 1\}, \quad i \in N^*, \; k \in B \tag{10}$$

$$x_i \in \mathbb{R}_{\geq 0}, \; y_i \in \mathbb{R}_{\geq 0}, \quad i \in N^* \tag{11}$$

$$L_{max} \in \mathbb{R}. \tag{12}$$

Eq. (1) defines the objective value. It minimizes the maximum lateness. Eq. (2) determines the relative position of any pair of items that are assigned to a same bin: one of them is either left of and/or below the other. Eq. (3) ensures that items i and j do not overlap horizontally if in the same bin while Eq. (4) guarantees that they do not overlap vertically. Eqs. (5) and (6) guarantee that i is entirely contained within a bin. Eq. (7) ensures either i or its rotated copy $i + n$ is packed into exactly one bin. Eq. (8) sets L_{max} larger than or equal to the lateness L_i of i, where L_i is the difference between the completion time of the bin to which i is assigned and the due date of i. Finally, Eqs. (9)–(12) declare the variables' types. The model has a quadratic number of variables in n. Because b is bounded by n, the model has a cubic number of constraints in n. The solution space has a large number of alternative solutions with many symmetric packing set ups. Subsequently, *EXACT* is hard to solve. Small-sized instances with as few as 20 items require significant computational effort.

3. Background

The two-dimensional orthogonal packing problem (2OPP) determines whether a set of rectangular items can be packed into a rectangular bin. This decision problem is used, in this paper, when generating the lower bound LB_3 (cf. Section 4.2) and as part of the new first-fit heuristic (*FF*) (cf. Section 5.1) when searching for a feasible packing.

LB_3 is the optimal solution of a mixed integer program that substitutes the containment and overlap constraints of *EXACT* by feasibility constraints. These constraints explore the notion of dual feasible functions (DFF) to strengthen the resulting relaxation of *EXACT*. Section 3.1 presents DFFs and explains their application to the non-oriented version of 2OPP.

2OPP arises also as a part of the constructive heuristic *FF*, which constitutes the first stage of the proposed solution approach *AP-PROX*. Specifically, every time it considers a subset of items, *FF* solves a non-oriented 2OPP to determine the feasibility of packing those items into a bin. As it calls the 2OPP decision problem several times, *FF* needs an effective way of tackling it. For this purpose, it models the problem as a CP, and augments it with two additional constraints issued from two related non-preemptive cumulative scheduling problems. Section 3.2 presents this CP model.

3.1. Feasibility constraints

Alves, Clautiaux, de Carvalho and Rietz (2016) explore standard DFFs for different combinatorial optimization problems, including cutting and packing problems. Fekete and Schepers (2004) apply DFFs to find a lower bound L_{2d} to the minimal number of bins needed to pack orthogonally a given set of two-dimensional oriented items. This section explains how to use DFFs to generate feasibility constraints.

A function $u: [0, 1] \rightarrow [0, 1]$ is dual feasible if

$$\sum_{s \in S} s \leq 1 \Rightarrow \sum_{s \in S} u(s) \leq 1$$

holds for any set S of non-negative real numbers. Let u_1 and u_2 be two valid DFFs. For the problem at hand, the DFFs transform the scaled sizes (w'_i, h'_i) of item $i \in I$ into differently scaled ones $(u_1(w'_i), u_2(h'_i)) \in (0, 1]$ where $w'_i = w_i/W$ and $h'_i = h_i/H$. For a feasible packing into a single bin to exist, the sum of the areas of the transformed items must be less than or equal to 1,

$$\sum_{i \in I} u_1(w'_i) u_2(h'_i) \leq 1. \tag{13}$$

This section explains how DFFs are combined in various ways to generate m inequalities/constraints in the form of Eq. (13) for the non-oriented 2OPP.

Let $\mathcal{A}^o = (\alpha^o_{ci}) \in R^{m \times n}_{\geq 0}$ and $\mathcal{A}^r = (\alpha^r_{ci}) \in R^{m \times n}_{\geq 0}$ denote two real-valued technological matrices. Element α^o_{ci} (resp. α^r_{ci}), $i \in N$, $c = 1, \ldots, m$, is a scaled area computed using w'_i and h'_i (resp. $w''_i = h_i/W$ and $h''_i = w_i/H$) as arguments for DFFs u_1 and u_2, respectively. Fekete and Schepers (2004) designed DFFs, namely $u^{(1)}$, $U^{(\epsilon)}$, and $\phi^{(\epsilon)}$, $\epsilon = p, q$. The functions and approach they use to obtained L_{2d} is used herein to derive the combinations of functions (u_1, u_2). The DFFs' input parameters $(p, q) \in (0, 0.5]^2$, as further specified in Section 6.

Furthermore, let $\mathbf{t}^o \in \{0, 1\}^n$ (resp. $\mathbf{t}^r \in \{0, 1\}^n$) be a binary decision vector such that $t^o_i = 1$ (resp. $t^r_i = 1$) if item $i, i \in N$, is packed into the bin without rotation (resp. with rotation) and 0 otherwise. When $t^o_i + t^r_i \leq 1$, the inequality

$$\sum_{i \in N} (\alpha^o_{ci} t^o_i + \alpha^r_{ci} t^r_i) \leq 1, \quad c = 1, \ldots, m, \tag{14}$$

derived from Eq. (13), is a valid feasibility constraint. Eq. (14) assumes that the rotated duplicate of an item $i, i \in N$, can fit into the bin. As mentioned in Section 2, when this assumption does not hold, $t^r_i = 0$ and is omitted from Eq. (14).

Some of the m constraints of Eq. (14) may be redundant. A constraint c, $c = 1, \ldots, m$, is redundant if either $\sum_{i \in N} \max(\alpha^o_{ci}, \alpha^r_{ci}) \leq 1$ or there exists c', $c' = 1, \ldots, m$, $c \neq c'$, such that both $\alpha^o_{ci} \leq \alpha^o_{c'i}$ and $\alpha^r_{ci} \leq \alpha^r_{c'i}$ for all $i \in N$.

3.2. Solving the 2OPP with constraint programming

This section develops a CP model for the non-oriented 2OPP. The CP model, which is an extension of the model of Clautiaux, Jouglet, Carlier and Moukrim (2008) for orthogonal packing, is strengthened by constraints issued of two non-preemptive cumulative scheduling problems. In this model, a bin corresponds to two distinct resources r_w and r_h of capacity W and H, respectively, while the items to two sets of activities $A^w = \{a^w_1, \ldots, a^w_{2n}\}$ and $A^h = \{a^h_1, \ldots, a^h_{2n}\}$ where a^w_i and a^h_i are the width and height of item $i, i \in N^*$. The first (resp. second) scheduling problem treats the widths (heights) of the items as processing times of activities A^w (resp. A^h) and considers the heights (widths) of the items as the amount of resource r_h (r_w) required to complete these activities. The activities in A^w and A^h have compatibility restrictions; i.e., $a^\bullet_i$ and $a^\bullet_{i+n}$, $\bullet = w, h$, cannot both be scheduled.

The first (resp. second) scheduling problem investigates whether its set of activities A^w (resp. A^h) can be performed within their respective time windows, without preemption and without exceeding the availability H (resp. W) of required resource r_h (resp. r_w). In fact, A^w and A^h are to be performed concurrently but using two different resources. Activity a^w_i has a processing time w_i and a time window $[0, W - w_i)$. To be processed, it uses an amount h_i of resource r_h. Similarly, activity a^h_i has a processing time h_i and a time window $[0, H - h_i)$. Its processing requires an amount w_i of resource r_w. Let s^w_i and s^h_i denote the respective starting times

of activities a_i^w and a_i^h. Then, s_i^w and s_i^h are the coordinates (x_i, y_i) of item i in the bin. The CP model that solves 2OPP is then given as:

$$\left(\text{PresenceOf}(a_i^w) \wedge \text{PresenceOf}(a_i^h)\right)$$
$$\neq \left(\text{PresenceOf}(a_{n+i}^w) \wedge \text{PresenceOf}(a_{n+i}^h)\right), \quad i \in N \quad (15)$$

$$\text{PresenceOf}(a_i^w) \neq \text{PresenceOf}(a_{n+i}^w), \quad i \in N \quad (16)$$

$$\text{PresenceOf}(a_i^h) \neq \text{PresenceOf}(a_{n+i}^h), \quad i \in N \quad (17)$$

$$\text{PresenceOf}(a_i^w) \wedge \text{PresenceOf}(a_i^h) \wedge \text{PresenceOf}(a_j^w) \wedge$$
$$\text{PresenceOf}(a_j^h) \Rightarrow \left(s_i^w + w_i \leq s_j^w\right) \vee \left(s_j^w + w_j \leq s_i^w\right)$$
$$\vee \left(s_i^h + h_i \leq s_j^h\right) \vee \left(s_j^h + h_j \leq s_i^h\right),$$
$$(i, j) \in N^{*2}, \quad i < j, \; j \neq n + i \quad (18)$$

$$\text{Cumulative}([s_1^w, \ldots, s_{2n}^w], [w_1, \ldots, w_{2n}], H) \quad (19)$$

$$\text{Cumulative}([s_1^h, \ldots, s_{2n}^h], [h_1, \ldots, h_{2n}], W). \quad (20)$$

Meta-constraint (15) guarantees that one of the pairs (a_i^w, a_i^h) and (a_{n+i}^w, a_{n+i}^h) is scheduled, where (a_i^w, a_i^h) and (a_{n+i}^w, a_{n+i}^h) correspond to item i and its rotated duplicate $n + i$. It uses the $\text{PresenceOf}(a)$ constraint that signals the presence of optional activity a, $a \in A^w \cup A^h$. It returns *true* when the optional activity a is present, and *false* otherwise. Constraint (16) forbids scheduling activity a_{n+i}^w when a_i^w is scheduled and vice versa. Similarly, constraint (17) prohibits scheduling activity a_{n+i}^h when a_i^h is scheduled. Despite the presence of constraint (15), constraints (16) and (17) are needed to eliminate some infeasible cases. For instance, constraint (15) discards neither the case where activities a_i^w, a_i^h, and a_{n+i}^w are scheduled while a_{n+i}^h is not nor the case where activities a_i^w, a_i^h, and a_{n+i}^h are scheduled while a_{n+i}^w is not. On the other hand, constraints (16) and (17) remove these cases. Constraint (18) ensures the no-overlap of any pair $(i, j) \in N^{*2}$, $i < j$, $j \neq n + i$, of packed items. Its left hand side holds when activities a_i^w, a_i^h, a_j^w, and a_j^h are scheduled and implies the right hand side, which is a disjunctive constraint that avoids the horizontal and vertical overlap of i and j by setting i to the left of j or i above j or j to the left of i or j above i, where the 'or' is inclusive. Finally, cumulative constraint (19) (resp. (20)) makes the activities of A^w (resp. A^h) complete within their respective time windows without exceeding the resource's capacity H (resp. W). Even though redundant, constraints (19) and (20) do strengthen the search. The CP model returns a feasible solution if and only if every item of N is assigned to the bin regardless of its rotation status.

For this model, the search tree is constructed as recommended by Clautiaux et al. (2008); i.e., variables $s_1^h, \ldots, s_{2n}^h$ are instantiated after variables $s_1^w, \ldots, s_{2n}^w$. The CP-model is solved via the IBM ILOG CP OPTIMIZER 12.6.2 (Laborie, 2009); set to the *restart mode*, which applies a failure-directed search when its large neighborhood search fails to identify an improving solution (Vilím, Laborie & Shaw, 2015). That is, instead of searching for a solution, it focuses on eliminating assignments that are most likely to fail. (cf. Laborie and Rogerie, 2008 and Vilím, 2009 for basics of optional interval variables (i.e. optional activities) and cumulative constraints.)

When allocated a threshold run time t_{lim}^{PACK}, the *CP Optimizer* acts as a heuristic, denoted hereafter as *PACK*. Preliminary experiments showed that *PACK* fathoms a large portion of infeasible solutions, especially when they are beyond "the edge of feasibility".

4. Lower bounds

This section presents three lower bounds for 2BPP with DD: two existing and a new one. These three bounds are compared in Section 6.2. Herein, LB designates the linear-time lower bound algorithm of Dell'Amico, Martello and Vigo (2002) for the non-oriented two-dimensional bin packing problem while $\underline{LB}(S)$ is a lower bound on the number of bins needed to pack the items of set S.

4.1. Existing lower bounds

The procedure to calculate LB_1 proceeds as follows. First, it sorts N in a non-decreasing order of the due dates, and sets $[j]$ to the item with the jth earliest due date. It then uses $\underline{LB}(S_{[j]})$, $j = 1, \ldots, n$, to deduce a lower bound of the lateness of the subset of items $S_{[j]} = \{[1], \ldots, [j]\}$. Some items must have a completion time $P \cdot \underline{LB}(S_{[j]})$; thus, have a lateness of at least $P \cdot \underline{LB}\left(S_{[j]}\right) - d_j$. Therefore, $LB_1 = \max_{j=1,\ldots,n}\{P \cdot \underline{LB}\left(S_{[j]}\right) - d_j\}$ is a valid lower bound on L_{max}.

Clautiaux, Jouglet and Hayek (2007) use DFFs to compute lower bounds for the non-oriented bin packing problem when the bin is a square. They show that their bounds dominate $\underline{LB}$ both theoretically and computationally for square bins. However, for rectangular bins, the quality of this bound remains an open issue.

LB_2, the second lower bound on L_{max}, is the result of the linear relaxation of *EXACT* where all the binary variables are substituted by variables in [0, 1].

4.2. A new lower bound

To the opposite of LB_2, which drops the integrality constraints, the new lower bound LB_3 is the *optimal* value of *RELAX*, which is a mixed integer programming relaxation of *EXACT*. *RELAX* exchanges the disjunctive constraints, given by Eqs. (2)–(6), with the feasibility constraints

$$\sum_{i \in N} \left(\alpha_{ci}^o f_{ik} + \alpha_{ci}^r f_{n+ik}\right) \leq 1, \; k \in B, \; c = 1, \ldots, m, \quad (21)$$

which are defined in the form of Eq. (14). The disjunctive constraints define the geometrical relationships between pairs of packed items and between a packed item and its assigned bin. They consider both the height and width of the items and bins and ensure the non-overlap of pairs of items and the containment of an item in the bin in both directions. The feasibility constraint, on the other hand, assimilates the item and the bin into dimension-less entities. Its inclusion in *RELAX* tightens the relaxation and improves the quality of the lower bound. Excluding it omits the layout aspect of the problem; thus, cannot produce reasonably good bounds. LB_3 is a valid bound if and only if *RELAX* is solved to optimality.

5. Approximate approaches

APPROX is a two-stage approximate approach for 2BPP with DD. The first stage constructs an initial solution and obtains related upper bounds using a new first-fit heuristic (*FF*). The second stage is iterative. It improves the current solution via an assignment-based heuristic *HEUR* and its relaxed version *HEUR′*, and updates the bounds if possible. The second stage diversifies its search when it stagnates. Sections 5.1–5.3 detail *FF*, *HEUR*, and *APPROX*.

5.1. First-fit heuristic

FF solves, via CP, a series of 2OPPs, where each 2OPP determines the feasibility of packing a given set of items into a single bin. It

Algorithm 1 First-fit heuristic algorithm FF(N).

1: sort N in a non-descending order of items' due dates;
2: set $k = 0$;
3: **while** ($N \neq \emptyset$) **do**
4:     set $k = k + 1$;
5:     open a new bin k setting $N_k = \emptyset$;
6:     **while** ($\underline{LB}(N_k) \leq 1$) **do**
7:         move the first item of N to N_k;
8:     **end while**
9:     **while** ($PACK(N_k) \Longleftrightarrow infeasible$) **do**
10:         move the last item of N_k back to its position in N;
11:     **end while**
12:     **for** each item $i \in N$ **do**
13:         **if** (($\underline{LB}(N_k \cup \{i\}) \leq 1$) **and** $PACK(N_k \cup \{i\}) \Longleftrightarrow feasible$)) **then**
14:             move item i from N to N_k;
15:         **end if**
16:     **end for**
17: **end while**
18: **return** solution as $(N_1, \ldots, N_k)$;

constructs a solution as detailed in Algorithm 1. It sorts the items of N in a non-descending order of their due dates, sets $k = 0$, and applies a sequential packing that iterates as follows until $N = \emptyset$. First, it determines the current bin k to be filled, and initializes its set N_k of packed items to the empty set. It removes the first item from N and inserts it into N_k. *Two scenarios are possible. When* $\underline{LB}(N_k) \leq 1$, it considers the next item of N. (The use of $\underline{LB}(N_k)$ ensures that *FF* starts with a dense packing; thus, limits the number of sequential calls to *PACK*.) *Otherwise*, it undertakes a backward step followed by an iterative sequential packing step. It calls *PACK* from Section 3.2 to determine whether it is possible to pack the items of N_k. When infeasibility is detected (potentially because *PACK* runs out of time), the backward step removes the last added item from N_k (because it may have caused the infeasibility) and inserts i back into N. Then it calls *PACK* again. When a feasible solution is obtained, *FF* proceeds with the iterative sequential packing step.

The constructive step considers the items of N sequentially. For every $i \in N$, it checks whether a feasible packing is possible. Specifically, it calls *PACK* when $\underline{LB}(N_k \cup \{i\}) \leq 1$. When *PACK* determines that it is possible to pack the items of $N_k \cup \{i\}$ into the current bin, the constructive step removes i from N and inserts it into N_k. Having tested all unpacked items of N, *FF* proceeds to the next bin by incrementing k to $(k + 1)$ if $N \neq \emptyset$. Hence, *FF* obtains an initial solution, characterized by its number of bins b and its corresponding maximum lateness *UB*. Subsequently, *APPROX* feeds this information to its second stage.

5.2. An assignment-based heuristic

The second stage of *APPROX* applies iteratively an assignment-based heuristic *HEUR*, which determines the feasibility of packing a set of oriented two-dimensional items into a set of multiple identical two-dimensional bins. Finding a feasible packing is hard not only because of the large number of alternative positions of an item within a bin but also because of the multitude of solutions having equal L_{max}. Herein, *HEUR* implements *four* strategies that enhance its performance. *First*, it reduces the search space to a subset of feasible positions, which correspond to the free regions within a bin. As it applies its greedy search to position items, it creates some free regions and fills others; thus, the search space is dynamic. *Second*, *HEUR* packs simultaneously as many items as possible into the various free regions. Thus, it reduces the

number of iterations needed to obtain a solution; consequently, it decreases its runtime. *Third*, *HEUR* implements a new kind of lookahead strategy that directs the search towards a feasible packing. This guiding mechanism imposes feasibility constraints that prohibit the current two-dimensional assignment problem *ASSIGN* from generating partial solutions that will lead to infeasible ones in future iterations. This innovative mechanism makes current decisions account for their impact on future ones. *Fourth* and last, *HEUR* uses upper bounds *UB* on L_{max} and $b(UB)$ on the number of bins. These bounds further reduce the search space: a candidate solution is a feasible packing whose $L_{max} < UB$ and which uses at most $b(UB)$ bins. Initially, *UB* is the L_{max} of the solution of *FF*.

HEUR, sketched in Algorithm 2, uses the following sets as input: the set N of not yet packed items, the set $\overline{N}$ of packed items, and the set E of available rectangular regions. These three sets are updated dynamically at each iteration. Initially, N is the set of the n items, $\overline{N} = \emptyset$, and $E = B$, where $|B| = b = b(UB)$. Thus, the set E_k of free regions contained in bin $k, k \in B$, is initially the kth bin: $E_k = \{(W, H)\}$, with $E_k \subseteq E$ and $\cup_{k \in B} E_k = E$.

Algorithm 2 Assignment-based heuristic *HEUR(UB)*.

1: initialize $\overline{N} = \emptyset$ and $E = B$, $|B| = b = b(UB)$;
2: **while** (true) **do**
3:     **if** ($ASSIGN(N, \overline{N}, E) \Longleftrightarrow feasible$) **then**
4:         **for** each item $i \in N$ **do**
5:             **if** ($\exists\, e \in E : (\varphi^o_{ie} = 1) \vee (\varphi^r_{ie} = 1)$) **then**
6:                 set $\rho^o_{ik} = \varphi^o_{ie}$ and $\rho^r_{ik} = \varphi^r_{ie}$ for $k : e \in E_k$;
7:                 move item i from N to $\overline{N}$;
8:             **end if**
9:         **end for**
10:         **if** ($N = \emptyset$) **then**
11:             return *feasible solution*;
12:         **end if**
13:         calculate coordinates (x'_i, y'_i) and (x''_i, y''_i) for every item $i \in \overline{N}$;
14:         update E exploring regions on top and to the right of every item $i \in \overline{N}$;
15:         **for** each region $e \in E_k, k \in B$ **do**
16:             **if** ($\nexists\, i \in N : (w_i \leq W_e) \wedge (h_i \leq H_e) \wedge (kP - d_i < UB)$) **then**
17:                 move e from E to $\overline{N}$;
18:             **end if**
19:         **end for**
20:         **if** ($\exists\, i \in N : E^o_i \cup E^r_i = \emptyset$) **then**
21:             return *infeasible solution*;
22:         **end if**
23:     **else**
24:         return *infeasible solution*;
25:     **end if**
26: **end while**

Let $(e, e') \in E^2$ denote two free regions characterized by their respective dimensions (W_e, H_e) and $(W_{e'}, H_{e'})$ and by their bottom leftmost coordinate positions (x_e, y_e) and $(x_{e'}, y_{e'})$ in their respective bins. When in a same bin, e and e' may overlap, as in Fig. 1. To guard against assigning two items to the overlap area of e and e', *HEUR* includes, into the assignment model, geometrical and disjunctive conditions that only apply if e and e' are in the same bin and overlap. *HEUR* signals such overlap via four parameters.

- $\theta^1_{ee'} = 1$ if $(x_e < x_{e'}) \wedge (y_e > y_{e'})$ as in Fig. 1(a) and 0 otherwise.
- $\theta^2_{ee'} = 1$ if $(x_e < x_{e'}) \wedge (y_e < y_{e'})$ as in Fig. 1(b) and 0 otherwise.
- $\theta^3_{ee'} = 1$ if $(x_e = x_{e'}) \wedge (y_e > y_{e'})$ as in Fig. 1(c) and 0 otherwise.
- $\theta^4_{ee'} = 1$ if $(x_e < x_{e'}) \wedge (y_e = y_{e'})$ as in Fig. 1(d) and 0 otherwise.

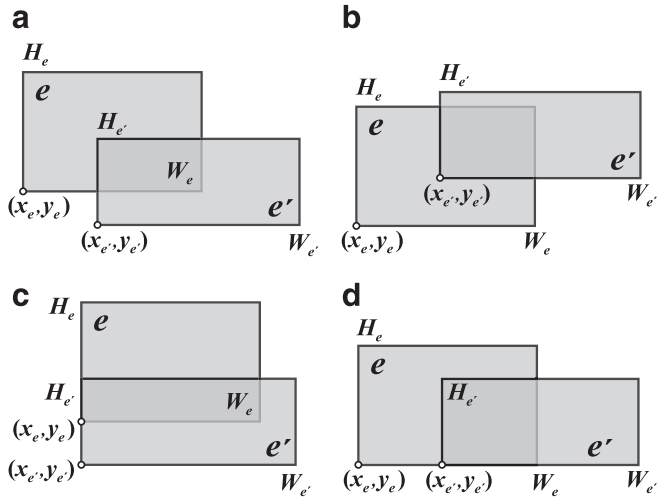Similarly, it signals the already packed items via parameters

Fig. 1. Patterns where two regions e and e' overlap.

- $\rho_{ik}^o = 1$ if $i \in \overline{N}$ is packed in $k \in B$ without rotation and 0 otherwise; and
- $\rho_{ik}^r = 1$ if $i \in \overline{N}$ is packed in $k \in B$ with rotation and 0 otherwise.

When item i is not yet packed (i.e., $i \in N$), $\sum_{k \in B} \rho_{ik}^o + \rho_{ik}^r = 0$.

Let $E_i^U = \cup_{k \in B} \{E_k : kP - d_i < UB\}$ denote the set of regions where $i \in N$ can be scheduled and $L_i < UB$. The set $E_i^U \supseteq E_i^o \cup E_i^r$, where $E_i^o = \cup_{e \in E_i^U} \{e : (w_i \le W_e) \wedge (h_i \le H_e)\}$ and $E_i^r = \cup_{e \in E_i^U} \{e : (h_i \le W_e) \wedge (w_i \le H_e)\}$ are the sets of regions where i can be positioned without and with rotation, respectively.

In each iteration, *HEUR* solves *ASSIGN*, which attaches a subset of unpacked items to regions of E using the following variables.

- $\varphi_{ie}^o = 1$ if item $i \in N$, is assigned without rotation to $e \in E_i^o$ and 0 otherwise.
- $\varphi_{ie}^r = 1$ if i is assigned with rotation to $e \in E_i^r$ and 0 otherwise.
- $f_{ik}^o = 1$ if i can be packed without rotation in a future iteration in bin k such that $L_i = Pk - d_i < UB$, and 0 otherwise.
- $f_{ik}^r = 1$ if i can be packed with rotation in a future iteration in bin k such that $L_i = Pk - d_i < UB$, and 0 otherwise. f_{ik}^o and f_{ik}^r allocate free space for items to be packed in future iterations without increasing UB.
- $w_e \in [0, W_e]$ and $h_e \in [0, H_e]$, the width and height of the used area of $e \in E$ when an item is positioned in (x_e, y_e).
- $l_{ee'}$ and $u_{ee'}$ are binary. They refer to the relative position of two areas e and e'. $l_{ee'} = 1$ (resp. $u_{ee'} = 1$) is used to make e to the left of (resp. below) e' when e and e' are part of the same bin.

ASSIGN maximizes the total profit generated by the packed items subject to non-overlap and containment constraints. The profit of $i, i \in N$, is $s_i \ge 0$. When $s_i = w_i h_i$, *ASSIGN* maximizes the utilization of the bins; i.e., the density of the packed items. Formally, *ASSIGN* is modeled as an MIP:

$$\max \quad \sum_{i \in N} s_i \left(\sum_{e \in E_i^o} \frac{\varphi_{ie}^o}{W_e H_e} + \sum_{e \in E_i^r} \frac{\varphi_{ie}^r}{W_e H_e} \right) \tag{22}$$

$$\text{s.t.} \quad \sum_{\substack{i \in N: \\ e \in E_i^o}} \varphi_{ie}^o + \sum_{\substack{i \in N: \\ e \in E_i^r}} \varphi_{ie}^r \le 1, \quad e \in E \tag{23}$$

$$\sum_{e \in E_i^o} \varphi_{ie}^o + \sum_{e \in E_i^r} \varphi_{ie}^r + \sum_{\substack{k \in B: \\ E_k \cap E_i^o \ne \emptyset}} f_{ik}^o + \sum_{\substack{k \in B: \\ E_k \cap E_i^r \ne \emptyset}} f_{ik}^r = 1, \quad i \in N \tag{24}$$

$$\sum_{e \in E_k} \left(\sum_{\substack{i \in N: \\ e \in E_i^o}} \alpha_{ci}^o \varphi_{ie}^o + \sum_{\substack{i \in N: \\ e \in E_i^r}} \alpha_{ci}^r \varphi_{ie}^r \right) + \sum_{\substack{i \in N: \\ E_k \cap E_i^o \ne \emptyset}} \alpha_{ci}^o f_{ik}^o + \sum_{\substack{i \in N: \\ E_k \cap E_i^r \ne \emptyset}} \alpha_{ci}^r f_{ik}^r \le 1$$
$$- \sum_{i \in \overline{N}} \left(\alpha_{ci}^o \rho_{ik}^o + \alpha_{ci}^r \rho_{ik}^r \right), \quad k \in B, \ c = 1, \dots, m \tag{25}$$

$$\sum_{\substack{i \in N: \\ e \in E_i^o}} w_i \varphi_{ie}^o + \sum_{\substack{i \in N: \\ e \in E_i^r}} h_i \varphi_{ie}^r \le w_e, \quad e \in E \tag{26}$$

$$\sum_{\substack{i \in N: \\ e \in E_i^o}} h_i \varphi_{ie}^o + \sum_{\substack{i \in N: \\ e \in E_i^r}} w_i \varphi_{ie}^r \le h_e, \quad e \in E \tag{27}$$

$$x_e + w_e \le x_{e'} + W_e(1 - l_{ee'}), \quad (e, e') \in E^2 : \theta_{ee'}^1 + \theta_{ee'}^2 = 1 \tag{28}$$

$$y_{e'} + h_{e'} \le y_e + H_{e'}(1 - u_{e'e}), \quad (e, e') \in E^2 : \theta_{ee'}^1 = 1 \tag{29}$$

$$y_e + h_e \le y_{e'} + H_e(1 - u_{ee'}), \quad (e, e') \in E^2 : \theta_{ee'}^2 = 1 \tag{30}$$

$$h_{e'} + (y_{e'} + H_{e'} - y_e) \left(\sum_{\substack{i \in N: \\ e \in E_i^o}} \varphi_{ie}^o + \sum_{\substack{i \in N: \\ e \in E_i^r}} \varphi_{ie}^r \right) \le H_{e'},$$
$$(e, e') \in E^2 : \theta_{ee'}^3 = 1 \tag{31}$$

$$w_e + (x_e + W_e - x_{e'}) \left(\sum_{\substack{i \in N: \\ e' \in E_i^o}} \varphi_{ie'}^o + \sum_{\substack{i \in N: \\ e' \in E_i^r}} \varphi_{ie'}^r \right) \le W_e,$$
$$(e, e') \in E^2 : \theta_{ee'}^4 = 1 \tag{32}$$

$$l_{ee'} + u_{e'e} \ge 1, \quad (e, e') \in E^2 : \theta_{ee'}^1 = 1 \tag{33}$$

$$l_{ee'} + u_{ee'} \ge 1, \quad (e, e') \in E^2 : \theta_{ee'}^2 = 1 \tag{34}$$

$$\varphi_{ie}^o \in \{0, 1\}, \ \varphi_{ie}^r \in \{0, 1\}, \quad i \in N, \ e \in E \tag{35}$$

$$f_{ik}^o \in \{0, 1\}, \quad i \in N, \ k \in B : E_k \cap E_i^o \ne \emptyset \tag{36}$$

$$f_{ik}^r \in \{0, 1\}, \quad i \in N, \ k \in B : E_k \cap E_i^r \ne \emptyset \tag{37}$$

$$l_{ee'} \in \{0, 1\}, \ u_{ee'} \in \{0, 1\},$$
$$(e, e') \in E^2 : \left(\theta_{ee'}^1 = 1 \right) \vee \left(\theta_{ee'}^2 = 1 \right) \tag{38}$$

$$0 \le w_e \le W_e, \ 0 \le h_e \le H_e, \quad e \in E. \tag{39}$$

Eq. (22) defines the objective function value as the weighted sum of the profits of packed items where the weight of an item i is inversely proportional to the area of region e used for its positioning. Eq. (23) prohibits packing more than one item into any region $e \in E$.

Eqs. (24) and (25) are part of the lookahead strategy. They employ f_{ik}^*, $* = o, r, i \in N, k \in B$, to reserve a free space for unpacked items. Eq. (24) assigns $i, i \in N$, either to one of the available regions during the current iteration or to one of the bins during a later iteration. Eq. (25) imposes the set of feasibility constraints. Here, $c, c = 1, \dots, m$ determines a vector of transformed areas (α_{ci}^o and α_{ci}^r) computed for all the items on $N \cup \overline{N}$ and their rotated copies and represented via matrices $\mathcal{A}^o$ and $\mathcal{A}^r$ (cf. Section 3.1). For every $c, c = 1, \dots, m$, and $k \in B$, Eq. (25) requires that the sum of

the transformed areas of (i) the items that have been previously packed ($\rho_{ik}^* = 1$, $* = o$, r), (ii) those being packed at the current iteration ($\varphi_{ie}^* = 1$, $* = o$, r), and (iii) those to be packed in future iterations ($f_{ik}^* = 1$, $* = o$, r) in selected bin k be bounded by 1. Even though it discards many partial solutions that lead to an infeasible packing, Eq. (25) does not guarantee that a not-yet-packed i will get a feasible position during later iterations.

Eqs. (26) and (27) determine w_e and h_e of the used area of e by imposing that w_i and h_i do not exceed w_e and h_e if i is assigned to e.

Eqs. (28)–(34) guarantee the non-overlap of a pair of items packed in two overlapping regions (e, e'). They substitute the full set of the disjunctive constraints that are traditionally used to ensure the non-overlap of packed items in a bin. This substitution reduces the number of constraints by eliminating redundant ones. That is, instead of considering all possible pairs of regions, *ASSIGN* focuses on those that can potentially create an overlap of packed items. It detects these regions via parameters $\theta_{ee'}^1$ to $\theta_{ee'}^4$.

Eqs. (28)–(30) focus on the case where e is to the left of e' but e and e' overlap. For those regions, Eq. (28) makes the x-coordinate of the rightmost point of the used area of e less than or equal to its counterpart for the leftmost point of e'. Eqs. (29) and (30) constrain the vertical positions of the used areas of e and e'. Eq. (29) deals with the case when e' is below e and $\theta_{ee'}^1 = 1$ as in Fig. 1(a). It restricts the y-coordinate of the topmost point of the used area of e' to be less than or equal to its counterpart of the bottommost point of e. Similarly, when e' is below e and $\theta_{ee'}^2 = 1$, Eq. (30) constrains the topmost y-coordinate of the used area of region e to be less than or equal to the lowest y-coordinate of region e'; thus avoiding the potential overlap of items assigned to the two regions depicted in Fig. 1(b).

Eqs. (31) and (32) deal with two special cases: the left sides of e and e' are aligned vertically, and the bottom sides of e and e' are aligned horizontally. When e and e' are aligned vertically and an item is packed in e, Eq. (31) constrains the topmost y-coordinate of the used area of region e' to be less than or equal to the lowest y-coordinate of region e as in Fig. 1(c). Similarly, when both e and e' are aligned horizontally and an item is positioned into e', Eq. (32) restricts the rightmost x-coordinate of the used area of region e to be less than or equal to the leftmost x-coordinate of e' as in Fig. 1(d).

Eqs. (33) and (34) ensure that the used areas of any pair of overlapping regions (e, e') are such that the used area of e' is below e, the used area of e is below e' or the used area of e is to the left of e'.

Finally, Eqs. (35)–(39) declare the types of the decision variables.

When *ASSIGN* returns a feasible solution, *HEUR* moves the packed items from N to $\overline{N}$, and sets the parameters $\rho_{ik}^* = 1$, $* = o$, r, $i \in \overline{N}$, $k \in B$, of the items packed in the current iteration. Next, it calculates the coordinates (x_i', y_i') and (x_i'', y_i'') of both the upper left and the bottom right corners of item $i \in \overline{N}$, where $(x_i', y_i') = (x_i, y_i + h_i \rho_{ik}^o + w_i \rho_{ik}^r)$ and $(x_i'', y_i'') = (x_i + w_i \rho_{ik}^o + h_i \rho_{ik}^r, y_i)$. Finally, *HEUR* updates E using the following two-step approach.

The first step defines the region $e^t = (W_{e^t}, H_{e^t})$ on top of item i (cf. Fig. 2(a)). To identify the height H_{e^t}, it searches along the ray $x = x_i$ and $y \geq y_i'$ for the first bottom side of another item j if such an item exists or the upper side of the bin. It sets $H_{e^t} = y^t - y_i'$, where $y = y^t$ is the line intersecting this side, with $y^t = y_j$ if j exists, and $y^t = H$ otherwise. To determine the width W_{e^t}, it expands its search along the line $y = y_i'$; i.e., to both the left and right sides of $x = x_i$. It shifts the left edge of e^t until it meets the first right edge of an item a or the left border of the bin. It determines the line $x = x^\ell$ intersecting this side where $x^\ell = x_a''$ if a exists, and
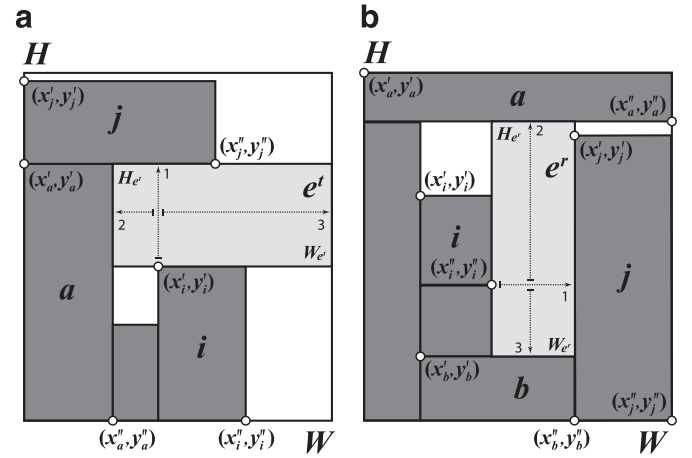
$x^\ell = 0$ otherwise. Similarly, it moves the right edge of e^t until it meets either the first left edge of an item b or the right border of the bin. It finds the line $x = x^r$ intersecting this side where $x^r = x_b$ if b exists, and $x^r = W$ otherwise. Subsequently, $W_{e^t} = x^r - x^l$.

The second step defines the region $e^r = (W_{e^r}, H_{e^r})$ to the right of i (cf. Fig. 2(b)). It identifies the width W_{e^r} by searching along the ray $y = y_i$, $x \geq x_i''$ for the first left side of another item j if j exists or the right side of the bin. It finds the line $x = x^r$ intersecting this side where $x^r = x_j$ if j exists and $x^r = W$ otherwise. It deduces $W_{e^r} = x^r - x_i''$. It determine the height H_{e^r} by expanding its search along the line $x = x_i''$; i.e., above and below $y = y_i$. It moves the top edge of e^r until it meets either the first bottom edge of an item a or the top border of the bin. It sets the line intersecting this side to $y = y^t$ where $y^t = y_a'$ if a exists, and $y^t = H$ otherwise. Similarly, it shifts the bottom edge of e^r until it meets the first top edge of an item b or the lower border of the bin. It finds the line $y = y^b$ intersecting this side where $y^b = y_b'$ if b exists, and $y^b = 0$ otherwise. Subsequently, $H_{e^r} = y^t - y^b$.

Having defined e^t and e^r, *HEUR* examines their "utility". It discards e if e cannot hold at least one of the unpacked items of N or e can hold an unpacked item but yields a lateness that is larger than or equal to *UB*. *HEUR* inserts a discarded region e into $\overline{N}$ treating e as a dummy item packed in bin k. This insertion strengthens Eq. (25). Finally, *HEUR* inserts all non-discarded regions into E, and checks whether the stopping criterion is satisfied. In fact, *HEUR* stops when all the items are packed or there is an unpacked item that does not fit into any region of E. When the stopping criterion is not met, *HEUR* runs another iteration with the updated N and $\overline{N}$.

5.3. Solution process as a whole

APPROX, detailed in Algorithm 3, consists of two stages. The first stage applies *FF* to obtain an initial feasible solution to 2BPP with DD along with an upper bound *UB* on L_{max} and an upper bound $b(UB)$ on the number of bins in an optimal solution. The second stage strives to improve these two bounds and the current solution using both the assignment-based heuristic *HEUR* and its relaxed version *HEUR'*. Specifically, it solves the non-oriented 2OPP with $b = b(UB) = \max_{i \in N} \{ \lfloor (UB + d_i)/P \rfloor \}$ bins so that the maximal lateness of any feasible solution is less than *UB*. It solves the problem in two steps, each consisting of two loops: an outer loop whose objective is to identify a solution with a tighter *UB* rapidly and an inner loop whose objective is to refine the search.

In the first step, the outer loop (cf. lines 5–20 of Algorithm 3) resets the profits $s_i = w_i h_i$, $i \in N$, and the iteration counter *count*

Fig. 2. New regions created by the partial packing of a bin.

Algorithm 3 The framework of *APPROX*.

1: **Stage 1: Initialization**
2: run *FF* to obtain an initial solution;
3: compute *UB* and $b = b(UB) = \max_{i \in N} \{\lfloor (UB + d_i)/P \rfloor\}$ based on the solution of Line 2;
4: **Stage 2: Iterative Step**
5: **repeat**
6:     set $s_i = w_i h_i, \ i \in N$;
7:     set $count = 0$;
8:     **repeat**
9:         **if** (call $HEUR'(UB) \Longleftrightarrow feasible$) **then**
10:             compute *UB* and $b = b(UB)$ based on the solution of Line 9;
11:             break;
12:         **else**
13:             modify $s_i = \gamma w_i h_i, \ i \in N$, in Eq. (22) of *ASSIGN*;
14:             set $count = count + 1$;
15:         **end if**
16:     **until** $count \leq a_{lim}^{HEUR}$
17:     **if** $count > a_{lim}^{HEUR}$ **then**
18:         break;
19:     **end if**
20: **until** true
21: repeat lines 5–20 replacing *HEUR'* with *HEUR*;

to 1. Then the inner loop runs *HEUR'* (cf. lines 8–16), which is a reduced version of *HEUR* where Eq. (25) and the decision variables f_{ik}^o and f_{ik}^r, $i \in N$, $k \in B$, are omitted from the model *ASSIGN*. *HEUR'* is generally weaker than *HEUR* in terms of the tightness of the upper bound of lateness but is faster in terms of run time. When it obtains a feasible solution, *HEUR'* feeds *APPROX* with a solution whose $L_{max} < UB$; that is, it tightens *UB*. This feasible solution may also reduce $b = b(UB)$. Subsequently, *APPROX* exits the inner loop and runs one more iteration calling *HEUR'* again but this time with new values of *UB* and *b*.

On the other hand, when *HEUR'* fails to find a feasible solution, the inner loop diversifies the search by using a different set of random profits. It changes the profits to $s_i = \gamma w_i h_i, \ i \in N$, where γ is a random real from the continuous *Uniform*[1,3], and increments *count* by 1. If *count* is less than or equal to a maximal number of iterations a_{lim}^{HEUR}, the inner loop starts a new iteration by solving *HEUR'* with its modified profits in the objective function (i.e., in Eq. (22) of *ASSIGN*).

When *count* reaches the limit a_{lim}^{HEUR}, *APPROX* proceeds with the second step, which performs exactly the same actions as the first step does except that it applies *HEUR* instead of *HEUR'*. The use of *HEUR* should improve the search. Therefore, the first step presolves the problem quickly while the second looks for an enhanced solution.

Modifying the weight coefficients of Eq. (22) of *ASSIGN* is a random local search (*RLS*). The choice of this particular diversification strategy along with this specific range of γ was based on preliminary computational investigations. Tests have shown that *RLS* yields, on average, better results than evolutionary strategies and techniques such as the method of sequential value correction (Belov, Mukhacheva & Scheithauer, 2008). The superiority of *RLS* is due to the items' random order, which is further accentuated by the unequal weights. Classical approaches on the other hand do not tackle the highly symmetric nature of bin packing solutions. They mainly construct solutions based on the sequential packing of items in ascending order of their areas/widths/heights (Lodi et al., 2002).

6. Computational experiments

The objective of the computational investigation is fourfold. First, it compares the proposed lower bound LB_3 to both LB_1 and LB_2. Second, it assesses the quality of the solution values of *FF*, *APPROX* and *EXACT*. Third, it compares the performance of *FF* and *APPROX* to that of *MXGA*. Last, it evaluates the performance of *FF* and *APPROX* on large-sized instances. All comparisons apply the appropriate statistical tests. All inferences are made at a 5% significance level, and all confidence interval estimates have a 95% confidence level.

APPROX is implemented in C#, which evokes *IBM ILOG Optimization Studio 12.6.2* to handle MIP and CP models. It is run on a PC with a 4 gigabytes RAM and a 3.06 gigahertz Dual Core processor. The time limit t_{lim}^{PACK} for *PACK* is set to 2 seconds. This setting, inferred from preliminary computational investigations, gives the best tradeoff between density of packing and runtime. Indeed, a longer t_{lim}^{PACK} does not necessarily lead to better packing solutions while it unduly increases the runtime of *FF*. Similarly, a shorter t_{lim}^{PACK} often hinders *FF* from reaching a feasible packing; thus causes poor quality solutions. The maximal number of iterations for *count* is $a_{lim}^{HEUR} = 100$, which also represents the best trade-off between quality and performance of *APPROX* according to our earlier tests. Furthermore, $p \in \{0.15, 0.30, 0.45\}$, $q \in \{0.15, 0.30, 0.45\}$ and up to $m = 27$ feasibility constraints are generated for the model *ASSIGN* of Section 5.2. A larger number of constraints does not generally improve the solution quality but increases the runtime of *ASSIGN*. Despite their large variety, the feasibility constraints of *ASSIGN* do not always tighten the lower bound on the free space available for packing. Therefore, their larger number does not necessarily tighten the model.

Section 6.1 presents the benchmark set. Section 6.2 measures the tightness of LB_3. Section 6.3 assesses the performance of *FF*, *APPROX* and *EXACT* in terms of their optimality gaps and number of times they reach the optimum. Section 6.4 compares the results of *FF*, *APPROX* and *MXGA*. Finally, Section 6.5 studies the sensitivity of *FF*, *APPROX* and *EXACT* to problem size.

6.1. Computational set up

Bennell et al. (2013) generated the benchmark set (including the due dates) that we test. For each instance, they calculated LB_1, and applied their multi-crossover genetic algorithm *MXGA* to obtain upper bounds. (*MXGA* was coded in ANSI-C using Microsoft Visual C++ 6.0 and run on a Pentium 4, 2.0 gigahertz, 2.0 gigabytes RAM computer with a 120-second time limit per replication, and ten replications per instance.) They, then, computed the average percent deviations of their upper bounds from LB_1. In their paper, they reported these average deviations aggregated over problem size. We use their aggregated average deviations in the comparisons of Section 6.4. However, we recomputed their LB_1 for every instance to perform the comparisons of Sections 6.2 and 6.4.

Their benchmark set uses square bins ($W = H$) whose processing times $P = 100$. It consists of 10 categories as detailed in Table 1. Column 3 gives the width W of a bin. Column 4 specifies how items are generated. Each category is characterized by the dimensions of the items, with categories 1–6 having homogeneous items that are randomly generated from a specific discrete uniform whereas categories 7–10 contain heterogeneous items belonging to four types in various proportions. The four types correspond to items whose (w_i, h_i) are randomly selected from discrete uniforms on the respective ranges:

- type 1: $\left(\left[\frac{2}{3}W, W \right], \left[1, \frac{1}{2}W \right] \right)$;
- type 2: $\left(\left[1, \frac{1}{2}W \right], \left[\frac{2}{3}W, W \right] \right)$;
- type 3: $\left(\left[\frac{1}{2}W, W \right], \left[\frac{1}{2}W, W \right] \right)$; and

Table 1
Generation of the widths and heights of items.

Category	Set	Bin size (W)	Item size (w_i, h_i)
1	$\mathcal{L}$	10	Uniformly random in [1, 10]
2	$\mathcal{S}$	30	Uniformly random in [1, 10]
3	$\mathcal{L}$	40	Uniformly random in [1, 35]
4	$\mathcal{S}$	100	Uniformly random in [1, 35]
5	$\mathcal{L}$	100	Uniformly random in [1, 100]
6	$\mathcal{S}$	300	Uniformly random in [1, 100]
7	$\mathcal{L}$	100	Type 1 with probability 70%; types 2–4 with probability 10% each
8	$\mathcal{L}$	100	Type 2 with probability 70%; types 1, 3, 4 with probability 10% each
9	$\mathcal{L}$	100	Type 3 with probability 70%; types 1, 2, 4 with probability 10% each
10	$\mathcal{S}$	100	Type 4 with probability 70%; types 1–3 with probability 10% each

- type 4: $\left(\left[1, \frac{1}{2}W\right], \left[1, \frac{1}{2}W\right]\right)$.

The categories can be divided, according to the relative size of the items, into two sets $\mathcal{L}$ and $\mathcal{S}$. Set $\mathcal{L}$, which contains instances with relatively large items, consists of categories 1, 3, 5, 7, 8, and 9. Set $\mathcal{S}$, which contains instances with small items, consists of categories 2, 4, 6, and 10.

For each category, there are five problem sizes: $n = 20, 40, 60, 80,$ and 100, and ten instances per category and problem size. For each problem, there are three classes A, B, and C of due dates, generated from the discrete $Uniform[101, \beta P \cdot \underline{LB}]$ where $\beta = 0.6, 0.8,$ and 1.0; thus, a total of 1500 instances.

6.2. Quality of the lower bounds

This section compares the performance of LB_1, LB_2, and LB_3, where LB_1 is computed via the algorithm of Section 4.1, LB_2 is the value of the incumbent returned by *CPLEX* and LB_3 is the optimal value of *RELAX* when *CPLEX* identifies the optimum within 1 hour of runtime. Table 2 summarizes the statistics of the lower bounds per class, category, and problem size. It displays

- $\gamma_\bullet$, the percent deviation of $LB_\bullet$, $\bullet = 1, 2, 3$, from the tightest lower bound LB^* where $LB^* = \max\{LB_1, LB_2, LB_3, EXACT\}$, and $\gamma_\bullet = 100(LB^* - LB_\bullet)/LB^*$, with *EXACT* included in the computation of LB^* only when *EXACT* is proven optimal;
- $\eta_\bullet$, the number of times $LB_\bullet = LB^*$, $\bullet = 1, 2, 3$; and
- #, the number of times LB_3 is not a valid bound; i.e., the number of times the linear programming solver *CPLEX* fails to prove the optimality of its incumbent within the 1 hour time limit.

Table 3 reports statistics of the runtime of LB_3 along with the tallied # per class, category and problem size. The statistics of the runtime are the average RT, median Q_2, minimum $\underline{RT}$ and maximum $\overline{RT}$; all in seconds. The median (i.e., the 50th percentile) separates the ordered data into two parts with equal numbers of observations. It is a more appropriate measure of central tendency in the presence of outliers or when the distribution of the data is skewed.

The *Category* rows of Tables 2 and 3 display the same statistics as the tables but per category per class. Their last rows report these statistics per class. Finally, their last eight columns give the statistics over all classes. A missing value indicates that all ten instances are unsolved by LB_3; i.e., # = 10.

The analysis of Tables 2 and 3 suggests the following. LB_1 is the best lower bound in 1083 instances out of 1500 instances. Over all instances, its average deviation from LB^* is 5.7%. Its runtime is very reduced.

LB_2 never outperforms LB_3 nor LB_1. It matches LB^* for only 111 instances; i.e., in 7.40% of the cases. These instances have $n = 20$

and 40, and belong to categories 2, 4, and 6. The average percent deviation of LB_2 from LB^* is 72.5%.

LB_3 is a valid bound for 1244 instances. Its average runtime is 74.91 seconds. Its much smaller median (2.65 seconds) signals the existence of some outlier cases that increased the mean. This is expected since *CPLEX* is allocated up to one hour to prove the optimality of its incumbent.

For those 1244 instances, LB_3 enhances LB_1 for 361 out of 1500 instances; i.e., in 24.07% cases. Its average enhancement over these 361 instances is 31.30%. In addition, it matches LB_1 for another 463 instances; i.e., in 30.87% cases. Subsequently, it is the best lower bound (among LB_3, LB_2, LB_1) in 824 cases.

Fig. 3 displays the box plots of the percent deviations $\gamma_\bullet$ of $LB_\bullet$, $\bullet = 1, 2, 3$, from LB^* as a function of the class, size, category and set of the instances. A box plot reflects the central tendency, spread, and skewness of the observed values. Its box corresponds to the 25th, 50th and 75th percentiles whereas its fences extend to the lowest and largest value of the data. Its stars signal outliers or unusual observations. Fig. 3 infers that LB_3 is mostly superior for set $\mathcal{L}$; that is, for all sizes of categories 1, 3, 5, 9 and for small-sized instances ($n = 20$ and 40) of categories 7 and 8. Overall, the average percent deviations of LB_3 from LB_1 and from LB^* are 3.58% and 14.1%. Furthermore, LB_3 strictly dominates LB_2 in 1074 cases. The three quartiles of the percent improvement, over all instances, are: 146.4, 218.9 and 417.1; implying a larger enhancement for the cases with strict dominance. That is, LB_3 is at least one order of magnitude larger than LB_2 in most instances.

6.3. Quality of the new upper bounds

Table 4 displays for each class, category, and problem size, the average percent deviation $\delta^F, \delta^A, \delta^E$, of the upper bounds $L_{max}^F, L_{max}^A, L_{max}^E$, obtained, respectively, by *FF*, *APPROX* and *EXACT*, from the best known lower bound LB^*. In addition, Table 4 displays $\eta^\bullet$, the number of times $L_{max}^\bullet = LB^*$, $\bullet = F, A, E$.

Table 4 infers the following results. The mean $L_{max}^\bullet$, $\bullet = F, A, E$, is equal for both classes A and B and larger for class C. The average δ^F and the average δ^A are larger than their respective medians (i.e., 66.85% and 33.62% versus 37.91% and 9.89%, respectively); signaling few outliers that are enlarging the true size of δ^F and δ^A. This is expected from $\mathcal{NP}$-hard problems.

The mean δ^A is the smallest. Its point and confidence interval estimates are 33.62% and (28.57%, 38.67%). That is, on average, the application of the second phase of the algorithm improves the solution of *FF* (except when $L_{max}^F = L_{max}^A = L_{max}^*$). The mean improvement is of the order of 33.23%, with a 29.59% lower side estimate. On the other hand, the mean δ^E is the largest because *EXACT* fails to obtain reasonably good solutions for large instances.

There is no correlation between $\delta^F, \delta^A, \eta^F, \eta^A$ and n, but there is a moderate correlation between both δ^E, η^E and n with respective 0.402 and −0.598 Pearson correlation coefficients. This infers that as the problem size increases, *EXACT* may encounter increasing difficulty in getting the tightest upper bound. δ^E depends on the problem category. Its mean for categories 2, 4, and 6 are larger than those for the other categories. Similarly, η^E is category dependent. Its estimate is largest for category 9 and smallest for category 2. Finally, δ^A is category dependent. Its mean δ^A is smallest for categories 2 and 9 and largest for categories 7, 8, and 10. *APPROX* solves many of the instances of categories 2 and 9 to optimality. It neither reaches the optimum nor proves the optimality of its solutions for any of the instances of categories 7 and 8. Even though category dependent, the mean number of times optimality is proven does not differ among sets or classes.

A valid upper bound to L_{max} is the minimum of L_{max}^A and L_{max}^E. This upper bound equals LB^* for 586 out of 1500 instances; that is, in 39.07% of the cases.

Table 2
Comparison of lower bounds.

Category	n	Class A							Class B							Class C							All classes						
		γ_1	γ_2	γ_3	η_1	η_2	η_3	#	γ_1	γ_2	γ_3	η_1	η_2	η_3	#	γ_1	γ_2	γ_3	η_1	η_2	η_3	#	γ_1	γ_2	γ_3	η_1	η_2	η_3	#
1	20	6.3	65.9	0.6	4	0	8	0	7.7	78.4	3.0	4	0	6	0	14.0	85.8	0.8	4	0	7	0	9.3	76.7	1.5	12	0	21	0
1	40	5.0	59.1	0.0	5	0	9	1	8.8	68.9	0.0	3	0	9	0	24.3	85.2	0.0	2	0	10	0	12.7	71.1	0.0	10	0	28	1
1	60	5.3	55.4	0.0	5	0	9	1	13.6	60.8	0.0	2	0	9	1	21.6	81.6	0.0	3	0	9	1	13.5	66.0	0.0	10	0	27	3
1	80	9.3	52.5	0.0	2	0	9	1	14.6	57.6	0.0	2	0	8	2	28.0	83.5	0.0	3	0	8	2	17.3	64.5	0.0	7	0	25	5
1	100	1.7	50.9	0.0	8	0	4	6	3.6	59.7	0.0	6	0	5	5	16.0	82.8	0.0	3	0	7	3	7.1	64.5	0.0	17	0	16	14
Category 1		**5.5**	**56.8**	**0.2**	**24**	**0**	**39**	**9**	**9.7**	**65.1**	**0.7**	**17**	**0**	**37**	**8**	**20.8**	**83.8**	**0.2**	**15**	**0**	**41**	**6**	**12.0**	**68.5**	**0.4**	**56**	**0**	**117**	**23**
2	20	0.0	0.0	0.0	10	10	10	0	0.0	0.0	0.0	10	10	10	0	0.0	0.0	0.0	10	10	10	0	0.0	0.0	0.0	30	30	30	0
2	40	0.0	84.3	41.5	10	1	2	0	0.0	89.8	46.4	10	1	1	0	0.0	90.0	55.4	10	1	1	0	0.0	88.1	47.7	30	3	4	0
2	60	0.0	87.3	29.1	10	0	0	0	0.0	98.6	33.3	10	0	0	0	0.0	100.0	60.6	10	0	0	0	0.0	95.3	41.0	30	0	0	0
2	80	0.0	82.4	11.8	10	0	0	0	0.0	97.0	21.3	10	0	0	0	0.0	100.0	44.0	10	0	0	0	0.0	93.1	25.7	30	0	0	0
2	100	0.0	87.0	34.2	10	0	0	0	0.0	99.8	50.0	10	0	0	0	0.0	100.0	75.3	10	0	0	0	0.0	95.6	53.2	30	0	0	0
Category 2		**0.0**	**68.2**	**23.3**	**50**	**11**	**12**	**0**	**0.0**	**77.1**	**30.2**	**50**	**11**	**11**	**0**	**0.0**	**78.0**	**47.0**	**50**	**11**	**11**	**0**	**0.0**	**74.4**	**33.5**	**150**	**33**	**34**	**0**
3	20	4.7	75.1	7.4	5	0	7	0	7.0	89.7	3.5	7	0	6	0	20.9	94.9	11.2	4	0	5	0	10.9	86.5	7.4	16	0	18	0
3	40	8.2	63.9	0.9	5	0	9	0	12.1	79.8	0.4	5	0	8	0	9.5	89.1	7.7	6	0	6	0	9.9	77.6	3.0	16	0	23	0
3	60	8.1	60.5	0.0	3	0	9	1	11.2	69.4	0.1	4	0	9	0	23.7	84.7	0.0	2	0	9	1	14.3	71.5	0.1	9	0	27	2
3	80	4.2	58.6	0.0	5	0	7	3	4.2	65.3	0.0	7	0	4	6	13.3	78.3	0.0	6	0	7	3	7.2	67.4	0.0	18	0	18	12
3	100	3.1	53.5	0.0	6	0	4	6	5.1	63.1	0.0	6	0	5	5	13.7	90.2	0.0	6	0	5	5	7.3	68.9	0.0	18	0	14	16
Category 3		**5.7**	**62.3**	**2.1**	**24**	**0**	**36**	**10**	**7.9**	**73.5**	**1.0**	**29**	**0**	**32**	**11**	**16.2**	**87.4**	**4.6**	**24**	**0**	**32**	**9**	**9.9**	**74.4**	**2.6**	**77**	**0**	**100**	**30**
4	20	0.0	0.0	0.0	10	10	10	0	0.0	0.0	0.0	10	10	10	0	0.0	0.0	0.0	10	10	10	0	0.0	0.0	0.0	30	30	30	0
4	40	0.0	84.4	50.3	10	1	3	0	0.0	89.7	51.6	10	1	2	0	0.0	90.0	65.3	10	1	1	0	0.0	88.0	55.8	30	3	6	0
4	60	0.0	84.2	20.5	10	0	0	0	0.0	96.4	23.7	10	0	0	0	0.0	100.0	48.1	10	0	1	0	0.0	93.5	30.8	30	0	1	0
4	80	0.0	82.1	17.0	10	0	0	0	0.0	97.3	22.6	10	0	0	0	0.0	100.0	57.9	10	0	0	0	0.0	93.1	32.5	30	0	0	0
4	100	0.0	84.9	25.7	10	0	0	0	0.0	98.5	45.4	10	0	0	0	0.0	100.0	72.1	10	0	0	0	0.0	94.5	47.7	30	0	0	0
Category 4		**0.0**	**67.1**	**22.7**	**50**	**11**	**13**	**0**	**0.0**	**76.4**	**28.7**	**50**	**11**	**12**	**0**	**0.0**	**78.0**	**48.7**	**50**	**11**	**12**	**0**	**0.0**	**73.8**	**33.4**	**150**	**33**	**37**	**0**
5	20	6.1	69.7	1.9	5	0	7	0	16.4	85.1	7.8	5	0	6	0	20.9	92.1	7.7	5	0	6	0	14.5	82.3	5.8	15	0	19	0
5	40	7.4	58.1	0.0	4	0	9	1	7.0	71.6	0.2	5	0	8	1	23.9	91.4	3.4	2	0	9	0	12.8	73.7	1.3	11	0	26	2
5	60	4.9	62.4	0.0	5	0	7	3	1.4	65.3	0.0	8	0	7	3	20.1	82.0	0.0	5	0	10	0	8.8	69.9	0.0	18	0	24	6
5	80	5.7	54.1	0.0	5	0	6	4	8.7	62.6	0.0	6	0	5	5	19.7	75.9	0.0	3	0	8	2	11.4	64.2	0.0	14	0	19	11
5	100	1.7	56.0	0.0	7	0	3	7	0.0	55.0	0.0	10	0	1	9	18.6	76.9	0.0	3	0	8	2	6.8	62.6	0.0	20	0	12	18
Category 5		**5.2**	**60.1**	**0.5**	**26**	**0**	**32**	**15**	**6.7**	**67.9**	**2.5**	**34**	**0**	**27**	**18**	**20.6**	**83.7**	**2.4**	**18**	**0**	**41**	**4**	**10.8**	**70.5**	**1.9**	**78**	**0**	**100**	**37**

Table 2 (*continued*)

Category	n	Class A							Class B							Class C							All classes						
		γ_1	γ_2	γ_3	η_1	η_2	η_3	#	γ_1	γ_2	γ_3	η_1	η_2	η_3	#	γ_1	γ_2	γ_3	η_1	η_2	η_3	#	γ_1	γ_2	γ_3	η_1	η_2	η_3	#
6	20	0.0	0.0	0.0	10	10	10	0	0.0	0.0	0.0	10	10	10	0	0.0	0.0	0.0	10	10	10	0	0.0	0.0	0.0	30	30	30	0
6	40	0.0	49.2	20.8	10	5	5	0	0.0	50.0	21.3	10	5	6	0	0.0	50.0	29.7	10	5	5	0	0.0	49.7	23.9	30	15	16	0
6	60	0.0	87.2	8.4	10	0	0	0	0.0	98.9	17.3	10	0	0	0	0.0	100.0	46.2	10	0	0	0	0.0	95.4	24.0	30	0	0	0
6	80	0.0	93.9	42.4	10	0	0	0	0.0	100.0	60.0	10	0	0	0	0.0	100.0	83.8	10	0	0	0	0.0	98.0	62.1	30	0	0	0
6	100	0.0	87.3	23.8	10	0	0	0	0.0	99.4	38.7	10	0	0	0	0.0	100.0	73.2	10	0	0	0	0.0	95.6	45.2	30	0	0	0
Category 6		**0.0**	**63.5**	**19.1**	**50**	**15**	**15**	**0**	**0.0**	**69.7**	**27.5**	**50**	**15**	**16**	**0**	**0.0**	**70.0**	**46.6**	**50**	**15**	**15**	**0**	**0.0**	**67.7**	**31.0**	**150**	**45**	**46**	**0**
7	20	11.8	71.4	10.2	2	0	1	0	21.5	84.1	18.1	1	0	1	0	42.4	96.0	28.1	1	0	2	0	25.2	83.9	18.8	4	0	4	0
7	40	2.3	64.0	0.7	5	0	5	1	5.6	73.2	1.2	6	0	7	1	24.4	96.0	1.4	1	0	9	0	10.8	77.7	1.1	12	0	21	2
7	60	0.0	59.9	0.4	10	0	0	9	0.0	74.0	8.3	10	0	0	8	4.1	88.2	5.3	8	0	5	2	1.4	74.1	5.4	28	0	5	19
7	80	0.0	57.8	0.0	10	0	1	9	0.0	63.6		10	0		10	7.4	79.2	0.0	7	0	3	7	2.5	66.9	0.0	27	0	4	26
7	100	0.0	56.1		10	0		10	0.0	59.9		10	0		10	2.8	90.1	8.4	9	0	1	8	0.9	68.7	8.4	29	0	1	28
Category 7		**2.8**	**61.8**	**5.2**	**37**	**0**	**7**	**29**	**5.4**	**70.9**	**10.0**	**37**	**0**	**8**	**29**	**16.2**	**89.9**	**10.7**	**26**	**0**	**20**	**17**	**8.1**	**74.2**	**8.9**	**100**	**0**	**35**	**75**
8	20	17.6	71.0	15.6	1	0	1	0	20.2	82.0	20.6	1	0	1	0	47.7	95.9	28.4	0	0	1	0	28.5	83.0	21.5	2	0	3	0
8	40	0.0	59.2	1.0	10	0	7	2	5.9	75.3	0.0	4	0	8	2	12.6	93.0	0.0	5	0	10	0	6.2	75.8	0.3	19	0	25	4
8	60	0.0	64.4	4.2	10	0	1	6	0.7	74.5	0.0	9	0	3	7	5.4	90.0	16.7	7	0	5	4	2.1	76.3	9.0	26	0	9	17
8	80	0.0	55.6		10	0		10	0.0	57.6		10	0		10	1.3	93.9	0.0	9	0	2	8	0.4	69.1	0.0	29	0	2	28
8	100	0.0	55.8	5.1	10	0	0	9	0.0	63.5	0.0	10	0	1	9	0.5	92.1	0.0	9	0	1	9	0.2	70.5	1.7	29	0	2	27
Category 8		**3.5**	**61.2**	**8.1**	**41**	**0**	**9**	**27**	**5.3**	**70.6**	**9.4**	**34**	**0**	**13**	**28**	**13.5**	**93.0**	**13.2**	**30**	**0**	**19**	**21**	**7.5**	**74.9**	**10.5**	**105**	**0**	**41**	**76**
9	20	0.0	61.4	0.0	10	0	10	0	0.0	71.4	0.0	10	0	10	0	0.0	81.1	0.0	10	0	10	0	0.0	71.3	0.0	30	0	30	0
9	40	2.5	57.5	0.0	7	0	10	0	2.9	75.8	0.0	7	0	10	0	4.7	80.2	0.0	9	0	10	0	3.4	71.1	0.0	23	0	30	0
9	60	1.6	58.1	0.0	7	0	10	0	2.6	66.1	0.0	7	0	10	0	4.6	81.0	0.0	8	0	10	0	2.9	68.4	0.0	22	0	30	0
9	80	2.1	56.2	0.0	5	0	10	0	2.7	65.3	0.0	6	0	10	0	15.2	79.9	0.0	6	0	10	0	6.7	67.1	0.0	17	0	30	0
9	100	1.7	55.1	0.0	5	0	10	0	2.9	65.0	0.0	5	0	10	0	5.2	77.9	0.0	5	0	10	0	3.3	66.0	0.0	15	0	30	0
Category 9		**1.6**	**57.7**	**0.0**	**34**	**0**	**50**	**0**	**2.2**	**68.7**	**0.0**	**35**	**0**	**50**	**0**	**5.9**	**80.0**	**0.0**	**38**	**0**	**50**	**0**	**3.2**	**68.8**	**0.0**	**107**	**0**	**150**	**0**
10	20	9.4	72.5	7.8	5	0	6	0	15.0	80.1	13.9	4	0	4	0	13.0	96.5	3.4	6	0	7	0	12.5	83.1	8.4	15	0	17	0
10	40	3.0	68.5	5.0	8	0	5	0	2.6	75.9	1.4	7	0	7	0	8.6	91.8	13.2	7	0	6	0	4.8	78.7	6.5	22	0	18	0
10	60	0.0	64.9	4.6	10	0	4	1	5.7	76.0	6.9	6	0	4	1	15.3	98.7	6.5	6	0	5	0	7.0	79.9	6.0	22	0	13	2
10	80	0.0	60.8	3.4	10	0	4	3	0.7	68.1	2.6	8	0	5	1	10.3	98.0	4.1	7	0	6	1	3.7	75.7	3.3	25	0	15	5
10	100	0.4	56.8	3.1	9	0	3	3	0.6	66.7	4.7	8	0	3	4	3.2	92.0	10.8	9	0	4	1	1.4	71.8	6.7	26	0	10	8
Category 10		**2.6**	**64.7**	**5.0**	**42**	**0**	**22**	**7**	**5.0**	**73.4**	**6.1**	**33**	**0**	**23**	**6**	**10.1**	**95.4**	**7.6**	**35**	**0**	**28**	**2**	**5.9**	**77.8**	**6.3**	**110**	**0**	**73**	**15**
All categories		**2.7**	**62.3**	**9.6**	**378**	**37**	**235**	**97**	**4.2**	**71.3**	**12.9**	**369**	**37**	**229**	**100**	**10.3**	**83.9**	**19.3**	**336**	**37**	**269**	**59**	**5.7**	**72.5**	**14.1**	**1083**	**111**	**733**	**256**

Table 3

Statistics of the runtime of LB_3.

Category	n	Class A					Class B					Class C					All classes				
		RT	Q_2	$\underline{RT}$	$\overline{RT}$	#	RT	Q_2	$\underline{RT}$	$\overline{RT}$	#	RT	Q_2	$\underline{RT}$	$\overline{RT}$	#	RT	Q_2	$\underline{RT}$	$\overline{RT}$	#
1	20	2.14	1.86	0.22	4.74	0	2.04	1.97	0.29	5.01	0	1.99	1.90	0.27	4.11	0	2.05	1.90	0.22	5.01	0
1	40	8.87	4.92	0.65	24.24	1	32.87	5.65	0.62	255.06	0	10.11	4.32	0.47	39.24	0	17.57	4.61	0.47	255.06	1
1	60	53.65	16.54	4.88	313.27	1	51.58	12.94	3.86	239.62	1	79.90	14.31	6.81	452.46	1	61.71	16.19	3.86	452.46	3
1	80	124.08	51.46	15.52	450.12	1	539.04	41.87	15.25	3587.90	2	32.16	25.35	13.46	63.91	2	227.45	33.14	13.46	3587.90	5
1	100	609.61	532.60	95.66	1277.58	6	316.59	281.97	69.87	641.00	5	332.34	257.04	50.84	1129.49	3	396.74	300.79	50.84	1277.58	14
Category 1		**100.96**	**15.52**	**0.22**	**1277.58**	**9**	**159.73**	**12.46**	**0.29**	**3587.90**	**8**	**77.81**	**12.42**	**0.27**	**1129.49**	**6**	**112.37**	**13.46**	**0.22**	**3587.90**	**23**
2	20	0.01	0.01	0.01	0.01	0	0.01	0.01	0.01	0.01	0	0.01	0.01	0.01	0.02	0	0.01	0.01	0.01	0.02	0
2	40	0.92	0.50	0.01	2.95	0	0.82	0.61	0.01	1.94	0	0.74	0.10	0.01	2.32	0	0.82	0.10	0.01	2.95	0
2	60	0.24	0.23	0.22	0.29	0	0.22	0.22	0.19	0.25	0	0.42	0.20	0.15	2.43	0	0.29	0.23	0.15	2.43	0
2	80	1.33	0.64	0.44	5.31	0	1.33	0.70	0.34	4.49	0	1.79	1.61	0.39	4.30	0	1.48	1.02	0.34	5.31	0
2	100	1.49	0.98	0.53	3.63	0	1.78	1.00	0.72	4.54	0	0.94	0.71	0.51	2.51	0	1.40	0.93	0.51	4.54	0
Category 2		**0.80**	**0.36**	**0.01**	**5.31**	**0**	**0.83**	**0.30**	**0.01**	**4.54**	**0**	**0.78**	**0.31**	**0.01**	**4.30**	**0**	**0.80**	**0.32**	**0.01**	**5.31**	**0**
3	20	2.01	2.52	0.23	3.15	0	1.41	1.77	0.19	2.46	0	1.41	1.46	0.16	3.75	0	1.61	1.85	0.16	3.75	0
3	40	23.92	6.84	0.54	124.81	0	38.97	5.05	0.63	304.06	0	216.39	4.49	0.56	2127.61	0	93.09	5.50	0.54	2127.61	0
3	60	39.66	26.57	6.62	126.54	1	108.69	18.63	4.78	704.02	0	18.40	12.77	6.91	42.99	1	57.48	19.92	4.78	704.02	2
3	80	462.72	199.61	40.84	1750.29	3	100.84	103.68	27.72	168.29	6	58.55	34.68	18.30	209.24	3	225.13	72.54	18.30	1750.29	12
3	100	270.91	218.31	50.99	596.04	6	525.85	87.81	31.07	2352.10	5	54.94	54.79	27.86	79.38	5	284.83	73.95	27.86	2352.10	16
Category 3		**123.47**	**17.31**	**0.23**	**1750.29**	**10**	**115.98**	**12.78**	**0.19**	**2352.10**	**11**	**73.86**	**8.30**	**0.16**	**2127.61**	**9**	**104.09**	**10.72**	**0.16**	**2352.10**	**30**
4	20	0.01	0.01	0.01	0.01	0	0.01	0.01	0.01	0.01	0	0.01	0.01	0.01	0.01	0	0.01	0.01	0.01	0.01	0
4	40	0.44	0.02	0.01	1.46	0	0.38	0.02	0.01	1.66	0	0.38	0.02	0.01	1.53	0	0.40	0.02	0.01	1.66	0
4	60	0.24	0.23	0.21	0.28	0	0.23	0.23	0.20	0.27	0	0.21	0.21	0.15	0.25	0	0.22	0.23	0.15	0.28	0
4	80	0.84	0.53	0.39	1.44	0	1.44	1.60	0.37	2.76	0	1.65	1.57	0.35	4.09	0	1.31	1.41	0.35	4.09	0
4	100	1.22	1.01	0.74	3.56	0	2.71	2.06	0.51	6.26	0	1.93	0.89	0.48	5.00	0	1.95	0.98	0.48	6.26	0
Category 4		**0.55**	**0.25**	**0.01**	**3.56**	**0**	**0.95**	**0.25**	**0.01**	**6.26**	**0**	**0.83**	**0.24**	**0.01**	**5.00**	**0**	**0.78**	**0.25**	**0.01**	**6.26**	**0**
5	20	2.17	2.49	0.18	5.36	0	2.13	1.87	0.18	5.39	0	1.34	0.74	0.21	3.85	0	1.88	1.87	0.18	5.39	0
5	40	21.59	5.59	0.82	66.19	1	64.12	5.18	2.38	327.80	1	5.05	4.07	0.54	12.64	0	29.35	5.20	0.54	327.80	2
5	60	170.16	19.43	10.73	643.32	3	39.78	31.33	4.09	93.73	3	47.54	12.91	6.53	293.40	0	81.04	17.73	4.09	643.32	6
5	80	832.08	526.51	26.06	2395.61	4	138.64	101.81	25.94	359.81	5	209.72	64.21	16.76	892.81	2	387.55	101.81	16.76	2395.61	11
5	100	366.41	203.19	184.46	711.58	7	1436.20	1436.20	1436.20	1436.20	9	376.63	143.92	34.87	1883.37	2	462.37	193.83	34.87	1883.37	18
Category 5		**214.25**	**12.04**	**0.18**	**2395.61**	**15**	**93.94**	**6.53**	**0.18**	**1436.20**	**18**	**113.70**	**10.74**	**0.21**	**1883.37**	**4**	**139.25**	**9.98**	**0.18**	**2395.61**	**37**

(continued on next page)

Table 3 (*continued*)

| Category	n	Class A					Class B					Class C					All classes				
		RT	Q_2	$\underline{RT}$	$\overline{RT}$	#	RT	Q_2	$\underline{RT}$	$\overline{RT}$	#	RT	Q_2	$\underline{RT}$	$\overline{RT}$	#	RT	Q_2	$\underline{RT}$	$\overline{RT}$	#
6	20	0.01	0.01	0.01	0.01	0	0.01	0.01	0.01	0.01	0	0.01	0.01	0.01	0.02	0	0.01	0.01	0.01	0.02	0
6	40	0.69	0.02	0.01	2.78	0	0.37	0.02	0.01	1.26	0	0.24	0.03	0.01	1.05	0	0.43	0.02	0.01	2.78	0
6	60	0.22	0.22	0.19	0.24	0	0.21	0.21	0.15	0.25	0	0.19	0.20	0.15	0.24	0	0.21	0.21	0.15	0.25	0
6	80	0.77	0.42	0.34	2.06	0	1.26	0.90	0.29	3.33	0	0.58	0.32	0.26	2.87	0	0.87	0.40	0.26	3.33	0
6	100	0.73	0.70	0.65	0.85	0	2.74	3.68	0.59	4.57	0	0.94	0.56	0.47	3.98	0	1.47	0.70	0.47	4.57	0
Category 6		**0.48**	**0.23**	**0.01**	**2.78**	**0**	**0.91**	**0.22**	**0.01**	**4.57**	**0**	**0.39**	**0.21**	**0.01**	**3.98**	**0**	**0.60**	**0.23**	**0.01**	**4.57**	**0**
7	20	2.81	2.57	1.70	4.16	0	3.21	3.03	0.78	7.26	0	3.34	3.28	1.12	5.85	0	3.12	2.89	0.78	7.26	0
7	40	588.34	15.86	4.14	1995.20	1	492.04	11.88	3.42	3159.14	1	10.67	5.98	3.11	32.16	0	351.08	12.07	3.11	3159.14	2
7	60	23.84	23.84	23.84	23.84	9	330.31	330.31	65.48	595.14	8	293.33	28.11	5.76	2008.52	2	275.55	39.31	5.76	2008.52	19
7	80	119.79	119.79	119.79	119.79	9					10	897.89	30.61	29.59	2633.47	7	703.36	75.20	29.59	2633.47	26
7	100					10					10	2976.24	2976.24	2523.98	3428.51	8	2976.24	2976.24	2523.98	3428.51	28
Category 7		**260.32**	**4.16**	**1.70**	**1995.20**	**29**	**243.86**	**6.87**	**0.78**	**3159.14**	**29**	**337.36**	**6.33**	**1.12**	**3428.51**	**17**	**289.61**	**6.33**	**0.78**	**3428.51**	**75**
8	20	7.24	3.82	2.70	32.54	0	4.04	3.17	1.39	7.20	0	3.60	3.59	1.80	5.46	0	4.96	3.50	1.39	32.54	0
8	40	122.28	35.51	15.01	530.00	2	154.01	28.13	7.33	772.32	2	11.00	7.32	4.16	41.59	0	89.24	18.35	4.16	772.32	4
8	60	245.57	89.54	35.19	768.01	6	790.76	28.03	22.36	2321.90	7	109.60	49.69	9.82	412.88	4	308.63	47.38	9.82	2321.90	17
8	80					10					10	2917.53	2917.53	2397.57	3437.50	8	2917.53	2917.53	2397.57	3437.50	28
8	100	144.22	144.22	144.22	144.22	9	1479.94	1479.94	1479.94	1479.94	9	311.61	311.61	311.61	311.61	9	645.26	311.61	144.22	1479.94	27
Category 8		**94.66**	**18.78**	**2.70**	**768.01**	**27**	**232.94**	**10.91**	**1.39**	**2321.90**	**28**	**239.66**	**7.00**	**1.80**	**3437.50**	**21**	**192.60**	**9.67**	**1.39**	**3437.50**	**76**
9	20	0.63	0.23	0.21	3.00	0	0.51	0.27	0.20	1.70	0	0.55	0.24	0.18	1.68	0	0.57	0.24	0.18	3.00	0
9	40	3.69	2.49	1.03	12.64	0	3.04	1.99	0.86	11.66	0	1.48	1.24	0.65	2.61	0	2.74	1.79	0.65	12.64	0
9	60	15.33	13.71	5.64	37.08	0	15.28	12.44	6.30	32.53	0	6.94	5.68	3.81	13.84	0	12.52	10.98	3.81	37.08	0
9	80	58.46	51.81	15.29	139.71	0	44.20	42.25	19.79	71.89	0	19.56	18.59	8.25	33.52	0	40.74	32.51	8.25	139.71	0
9	100	198.30	113.16	0.00	592.43	0	146.86	125.91	75.34	305.05	0	54.46	42.40	26.82	110.12	0	133.21	94.07	0.00	592.43	0
Category 9		**55.28**	**12.52**	**0.00**	**592.43**	**0**	**41.98**	**12.56**	**0.20**	**305.05**	**0**	**16.60**	**5.68**	**0.18**	**110.12**	**0**	**37.95**	**10.98**	**0.00**	**592.43**	**0**
10	20	2.03	1.83	0.20	4.29	0	1.75	1.85	0.53	3.08	0	1.73	1.74	0.24	3.40	0	1.84	1.76	0.20	4.29	0
10	40	21.96	2.22	0.60	175.73	0	13.09	3.38	0.43	103.75	0	2.27	1.65	0.36	5.26	0	12.44	2.31	0.36	175.73	0
10	60	14.80	10.46	6.06	35.88	1	15.51	8.66	5.08	65.43	1	8.28	6.74	0.84	34.47	0	12.70	8.39	0.84	65.43	2
10	80	51.53	25.45	10.41	222.78	3	220.03	20.10	13.77	1812.68	1	25.27	18.90	4.69	57.86	1	102.74	20.10	4.69	1812.68	5
10	100	145.47	38.22	15.89	801.33	3	616.26	61.51	27.76	3424.50	4	70.61	26.18	11.65	424.24	1	243.24	36.50	11.65	3424.50	8
Category 10		**40.75**	**10.03**	**0.20**	**801.33**	**7**	**135.59**	**8.39**	**0.43**	**3424.50**	**6**	**20.53**	**5.02**	**0.24**	**424.24**	**2**	**64.47**	**7.29**	**0.20**	**3424.50**	**15**
All categories		**71.54**	**2.57**	**0.00**	**2395.61**	**97**	**81.71**	**2.58**	**0.01**	**3587.90**	**100**	**71.84**	**2.87**	**0.01**	**3437.49**	**59**	**74.91**	**2.65**	**0.00**	**3587.90**	**256**
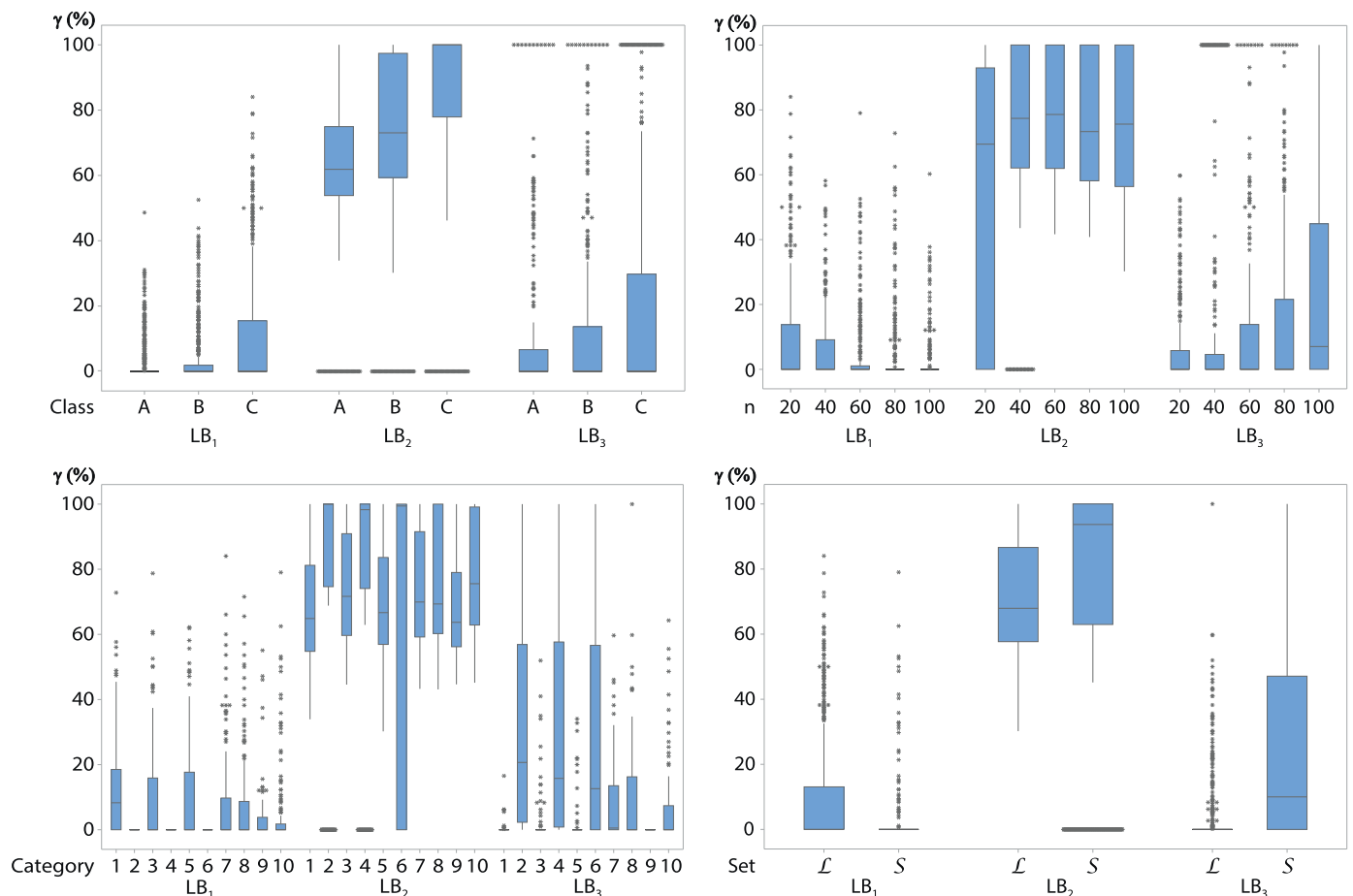
Fig. 3. Box plots for the mean percent deviations of the lower bounds from LB^*.

6.4. Comparing FF and APPROX to MXGA

This section investigates the performance of *APPROX* relative to existing upper bounds. Table 5 reports the results per class and category as Bennell et al. (2013) do not provide results per problem size. Column 3 gives the average number of attempts made by *HEUR* (and *HEUR'*) in order to reach a feasible solution within a single run. Columns 4–6 and 7–9 report statistics of the runtime $RT^\bullet$, $\bullet = F$, A, in seconds: the average, median, and maximum run time, all in seconds, over each set of 50 instances. RT^A includes RT^F as it pre-calls *FF*. RT^{MXGA} is fixed to 120 seconds per replication for each of the ten replications; thus, is not included in the table. Columns 10 and 11 give $\#^\bullet$, $\bullet = F$, A, the number of times the run times of *FF* and *HEUR* are larger than the 120 seconds runtime of one replication of *MXGA*. This number is out of 50 for each class and category and out of 500 for each class. Finally, Columns 12–14 report the average relative percent gap $\gamma^\bullet = 100(L_{max}^\bullet - LB_1)/LB_1$, $\bullet = F$, A, *MXGA*.

Table 5 shows that *HEUR* needs on average 23 attempts to reach a feasible solution. The mean number of attempts differs among categories. It is smaller for categories 2 and 9 with respective point estimates of 9.67 and 2.00. The instances of these two categories contain many tiny items. Therefore, packing these items is relatively easy. The mean number of attempts is larger for categories 7 and 8 with respective point estimates of 40.00 and 41.33. However, it is not different among sets $\mathcal{S}$ and $\mathcal{L}$; that is, what defines the level of difficulty of packing is the homogeneity of the items rather than their sizes. Finally, this mean number of attempts does not differ among classes. This is expected because the differences

of classes is caused by the tightness of the due dates and not by the packing procedure itself.

FF and *APPROX* identify a local optimum in 10.6 and 29.2 seconds, on average. These values are inflated by few outliers, as the box plots of Fig. 4 illustrate. In fact, the respective median run times are 1.2 and 15.6 seconds. Over all 1500 tested instances, *FF* needed a larger than 120-second runtime for 6 instances. Similarly, *APPROX* needed a larger than 120-second runtime for merely 63 out of 1500 instances; that is, in 4.2% of the tested cases. Fig. 5 displays the confidence interval estimates of the mean run times per category of *FF* and *APPROX*. The mean run times of both *FF* and *APPROX* are less than the 120-second runtime of *MXGA* at any level of significance. That is, *APPROX* is, on average, faster than *MXGA*.

FF solves the instances of set $\mathcal{L}$ particularly fast. However, it is relatively slower on the instances of set $\mathcal{S}$. In fact, the large number of items per bin for instances of $\mathcal{S}$ leads to inaccurate lower bounds $\underline{LB}$; thus, increases the number of calls to *PACK*, which checks the feasibility of a packing. In addition, this large number of items slows down *PACK*. This, in turn, translates into increased run times. *APPROX* spends, on average, approximately the same time to solve an instance of any category, except of categories 4 and 6.

FF is as competitive as *MXGA* in terms of solution quality. Their mean optimality gaps are not different. However, *FF* is faster than *MXGA*. In addition, *FF* outperforms *MXGA* on some categories of classes B and C.

APPROX outperforms *MXGA* on average. The mean optimality gap of *APPROX* is smaller than its *MXGA* counterpart, with point and confidence interval estimates of the mean difference of -27.45% and $(-44.22\%, -10.68\%)$. As further substantiated by

Table 4
Comparison of upper bounds.

Category	n	Class A δ^F	δ^A	δ^E	η^F	η^A	η^E	Class B δ^F	δ^A	δ^E	η^F	η^A	η^E	Class C δ^F	δ^A	δ^E	η^F	η^A	η^E	All classes δ^F	δ^A	δ^E	η^F	η^A	η^E
1	20	18.0	3.6	0.0	4	7	10	21.7	1.4	0.0	3	8	10	51.6	4.9	0.0	1	6	10	30.5	3.3	0.0	8	21	30
1	40	24.3	8.4	8.9	1	2	4	27.6	9.4	13.2	1	4	5	45.3	18.7	11.6	1	4	6	32.4	12.2	11.2	3	10	15
1	60	22.1	9.0	50.0	0	3	0	36.5	11.2	58.4	0	2	0	53.9	33.1	132.1	0	4	1	37.5	17.8	78.4	0	9	1
1	80	19.7	5.6	147.3	1	4	0	34.8	11.8	346.7	1	3	0	69.8	23.4	683.5	0	6	0	41.4	13.6	382.5	2	13	0
1	100	21.9	10.0	523.9	0	1	0	39.6	17.8	913.9	1	1	0	104.0	58.1	2147.0	0	1	0	55.2	28.6	1194.9	1	3	0
Category 1		**21.2**	**7.3**	**146.0**	**6**	**17**	**14**	**32.0**	**10.3**	**266.4**	**6**	**18**	**15**	**64.9**	**27.6**	**602.6**	**2**	**21**	**17**	**39.4**	**15.1**	**334.8**	**14**	**56**	**46**
2	20	0.0	0.0	0.0	10	10	10	2.2	0.0	0.0	9	10	10	1.7	0.0	0.0	8	10	10	1.3	0.0	0.0	27	30	30
2	40	6.4	4.7	7.9	5	8	0	10.6	3.7	8.5	6	8	0	33.7	4.2	28.7	4	8	1	16.9	4.2	15.0	15	24	1
2	60	44.6	1.3	406.4	2	5	0	19.3	5.0	610.5	2	3	0	42.7	11.6	1102.5	3	5	0	35.5	6.0	709.8	7	13	0
2	80	11.0	4.8	4132.6	3	5	0	8.4	1.6	4755.3	3	5	0	22.3	11.0	11144.7	2	4	0	13.9	5.8	6743.8	8	14	0
2	100	1.7	0.9	5579.3	3	5	0	4.6	4.1	9018.8	1	1	0	14.9	9.2	20412.2	1	2	0	7.1	4.7	11368.7	5	8	0
Category 2		**12.8**	**2.3**	**2025.2**	**23**	**33**	**10**	**9.0**	**2.9**	**2886.8**	**21**	**27**	**10**	**23.1**	**7.2**	**6254.5**	**18**	**29**	**11**	**14.9**	**4.1**	**3716.3**	**62**	**89**	**31**
3	20	39.6	9.5	2.7	1	4	9	66.9	23.0	8.3	2	5	9	63.7	8.4	0.0	3	6	10	56.7	13.7	3.7	6	15	28
3	40	28.2	15.4	14.6	1	2	3	50.0	29.5	29.6	0	2	3	63.3	36.8	39.8	1	3	4	47.2	27.3	28.0	2	7	10
3	60	37.1	13.7	153.1	0	1	0	52.3	22.0	130.5	0	1	0	88.4	45.1	166.5	0	1	0	59.3	26.9	150.1	0	3	0
3	80	47.6	17.5	446.6	0	0	0	82.5	44.7	982.9	0	0	0	131.7	74.4	778.4	0	2	0	87.3	45.5	736.0	0	2	0
3	100	38.4	17.8	873.6	0	1	0	60.2	30.3	1457.3	0	2	0	144.9	89.5	3003.7	0	0	0	81.1	45.8	1778.2	0	3	0
Category 3		**38.2**	**14.8**	**298.1**	**2**	**8**	**12**	**62.4**	**29.9**	**521.7**	**2**	**10**	**12**	**98.4**	**50.8**	**797.7**	**4**	**12**	**14**	**66.3**	**31.8**	**539.2**	**8**	**30**	**38**
4	20	19.6	0.0	0.0	6	10	10	5.2	0.0	0.0	8	10	10	2.8	0.0	0.0	7	10	10	9.2	0.0	0.0	21	30	30
4	40	12.1	0.6	1.7	1	5	3	30.1	6.7	12.3	1	2	0	341.4	54.4	83.9	2	5	2	127.9	20.5	32.6	4	12	5
4	60	77.2	23.1	471.7	0	2	0	78.7	27.3	514.0	0	1	0	130.0	39.0	755.0	0	2	0	95.3	29.8	584.0	0	5	0
4	80	50.5	17.6	1829.2	0	0	0	54.0	23.1	1812.3	0	0	0	96.2	33.6	8226.6	0	0	0	66.9	24.8	3808.8	0	0	0
4	100	24.3	11.1	5890.4	0	0	0	36.5	14.3	8423.9	0	0	0	129.5	50.9	21683.5	1	1	0	63.4	25.4	11665.3	1	1	0
Category 4		**36.8**	**10.5**	**1662.4**	**7**	**17**	**13**	**40.9**	**14.3**	**2152.5**	**9**	**13**	**10**	**140.0**	**35.6**	**5782.9**	**10**	**18**	**12**	**72.5**	**20.1**	**3174.6**	**26**	**48**	**35**
5	20	16.8	5.4	3.0	2	6	7	27.9	1.6	0.0	2	8	10	95.0	4.5	0.0	4	9	10	46.6	3.8	1.0	8	23	27
5	40	27.5	11.3	14.6	0	3	4	34.1	13.9	16.7	2	4	4	63.5	21.8	11.1	0	3	5	41.7	15.7	14.2	2	10	13
5	60	32.5	14.5	50.7	0	2	0	53.2	31.0	84.5	0	3	0	63.9	32.5	132.0	1	2	0	49.9	26.0	88.9	1	7	0
5	80	35.1	14.0	328.4	0	2	0	63.1	28.0	440.8	0	2	0	70.0	39.0	777.2	0	2	0	56.1	24.0	515.5	0	6	0
5	100	40.7	18.0	630.6	0	1	0	66.4	40.9	1141.8	0	0	0	158.5	90.5	2220.4	0	1	0	88.5	49.8	1330.9	0	2	0
Category 5		**30.5**	**12.6**	**208.6**	**2**	**14**	**11**	**48.9**	**23.1**	**336.8**	**4**	**17**	**14**	**90.2**	**35.9**	**638.3**	**5**	**17**	**15**	**56.5**	**23.9**	**394.2**	**11**	**48**	**40**

(*continued on next page*)

Table 4 (*continued*)

Category	n	Class A						Class B						Class C						All classes					
		δ^F	δ^A	δ^E	η^F	η^A	η^E	δ^F	δ^A	δ^E	η^F	η^A	η^E	δ^F	δ^A	δ^E	η^F	η^A	η^E	δ^F	δ^A	δ^E	η^F	η^A	η^E
6	20	36.7	0.0	0.0	3	10	10	19.0	0.0	0.0	3	10	10	8.1	0.0	0.0	4	10	10	21.3	0.0	0.0	10	30	30
6	40	65.2	18.1	19.2	0	2	1	45.0	13.5	16.1	0	2	1	230.8	61.8	150.5	0	2	1	113.7	31.1	61.9	0	6	3
6	60	102.4	11.5	625.0	0	4	0	141.0	15.2	657.5	0	0	0	399.3	218.9	3073.5	0	0	0	214.2	81.9	1396.1	0	4	0
6	80	61.6	1.7	2541.6	0	2	0	57.7	9.2	4408.7	0	0	0	409.3	46.8	17238.9	0	0	0	176.2	19.2	7746.7	0	2	0
6	100	69.0	15.4	6310.5	0	0	0	110.5	24.7	8378.1	0	0	0	192.7	57.2	16434.8	0	0	0	124.1	32.4	10374.4	0	0	0
Category 6		**67.0**	**9.3**	**1899.2**	**3**	**18**	**11**	**74.6**	**12.5**	**2692.1**	**3**	**12**	**11**	**248.0**	**76.9**	**7263.9**	**4**	**12**	**11**	**129.9**	**32.9**	**3907.0**	**10**	**42**	**33**
7	20	30.1	10.6	3.4	0	4	9	16.2	11.7	0.0	4	4	10	28.5	12.2	0.0	4	6	10	24.9	11.5	1.1	8	14	29
7	40	43.5	26.9	30.5	0	0	0	56.0	38.1	33.0	0	0	0	120.8	98.4	71.5	0	0	1	73.4	54.4	45.0	0	0	1
7	60	39.6	27.3	83.8	0	0	0	62.0	50.1	179.6	0	0	0	167.2	137.4	409.8	0	0	0	89.6	71.6	225.9	0	0	0
7	80	41.7	28.0	523.0	0	0	0	73.9	52.3	760.5	0	0	0	122.1	96.6	1837.6	0	0	0	79.2	59.0	1040.4	0	0	0
7	100	38.1	26.8	772.4	0	0	0	68.8	49.1	1324.1	0	0	0	235.4	209.6	4148.8	0	0	0	114.1	95.2	2081.8	0	0	0
Category 7		**38.6**	**23.9**	**282.6**	**0**	**4**	**9**	**55.4**	**40.3**	**465.1**	**4**	**4**	**10**	**134.8**	**110.8**	**1293.5**	**4**	**6**	**11**	**76.3**	**58.3**	**681.9**	**8**	**14**	**30**
8	20	16.5	3.8	0.1	0	4	9	12.8	3.2	0.0	3	6	10	67.3	55.7	0.0	0	2	10	32.2	20.9	0.0	3	12	29
8	40	43.4	28.6	30.5	0	0	0	59.3	42.0	38.4	0	0	0	136.3	96.3	83.3	0	0	1	79.6	55.6	50.7	0	0	1
8	60	45.4	24.9	97.6	0	0	0	60.5	41.8	130.7	0	0	0	423.6	294.0	824.1	0	0	0	176.5	120.2	359.5	0	0	0
8	80	40.9	25.6	566.6	0	0	0	72.7	54.4	847.4	0	0	0	399.2	307.5	4538.8	0	0	0	170.9	129.2	1984.3	0	0	0
8	100	38.6	25.4	684.0	0	0	0	72.6	51.9	1362.5	0	0	0	254.5	206.6	4570.2	0	0	0	121.9	94.6	2205.6	0	0	0
Category 8		**37.0**	**21.7**	**279.4**	**0**	**4**	**9**	**55.6**	**38.6**	**475.8**	**3**	**6**	**10**	**256.2**	**192.0**	**2003.3**	**0**	**2**	**11**	**116.2**	**84.1**	**923.8**	**3**	**12**	**30**
9	20	2.8	0.0	0.0	9	10	10	5.9	0.0	0.0	8	10	10	1.7	0.0	0.0	9	10	10	3.5	0.0	0.0	26	30	30
9	40	2.8	0.0	0.0	7	10	10	6.9	0.0	0.0	5	10	9	10.1	0.0	0.0	8	10	10	6.6	0.0	0.0	20	30	29
9	60	1.7	0.0	0.0	7	9	10	1.3	0.0	0.0	9	10	10	5.2	2.0	0.0	8	9	10	2.7	0.7	0.0	24	28	30
9	80	0.8	0.0	0.0	8	10	10	2.5	0.0	4.1	6	10	8	5.4	0.0	0.0	8	10	10	2.9	0.0	1.4	22	30	28
9	100	1.0	0.0	37.8	8	10	8	2.0	0.0	11.1	7	10	8	2.3	0.0	7.0	8	10	6	1.8	0.0	18.6	23	30	22
Category 9		**1.8**	**0.0**	**6.9**	**39**	**49**	**48**	**3.7**	**0.0**	**2.9**	**35**	**50**	**45**	**4.9**	**0.4**	**1.3**	**41**	**49**	**46**	**3.5**	**0.1**	**3.7**	**115**	**148**	**139**
10	20	25.4	13.4	12.5	3	6	7	37.9	26.4	16.2	2	6	8	229.0	165.4	161.1	1	4	7	97.4	68.4	63.3	6	16	22
10	40	29.9	14.2	22.1	0	1	0	50.2	33.2	39.8	0	1	1	209.0	153.6	172.3	1	4	4	96.4	67.0	78.0	1	6	5
10	60	32.4	17.0	214.4	0	0	0	52.0	36.5	166.6	0	0	0	86.3	66.1	645.2	0	0	0	56.9	39.8	330.4	0	0	0
10	80	31.3	18.7	397.9	0	0	0	54.3	35.4	1058.2	0	0	0	211.6	168.1	3409.9	0	0	0	99.0	74.1	1622.0	0	0	0
10	100	30.0	17.5	1295.2	0	0	0	47.2	32.3	1872.7	0	0	0	267.1	186.6	7995.1	0	0	0	114.8	78.8	3721.0	0	0	0
Category 10		**29.8**	**16.2**	**392.0**	**3**	**7**	**7**	**48.3**	**32.8**	**640.2**	**2**	**7**	**9**	**200.6**	**147.9**	**2553.0**	**2**	**8**	**11**	**92.9**	**65.6**	**1185.8**	**7**	**22**	**27**
All categories		**31.4**	**11.9**	**722.2**	**85**	**171**	**144**	**43.1**	**20.5**	**1042.8**	**89**	**164**	**146**	**126.1**	**68.5**	**2699.9**	**90**	**174**	**159**	**66.9**	**33.6**	**1483.7**	**264**	**509**	**449**

Table 5
Summary results of computational experiments.

| Class	Category	Iter	RT^F			RT^A			$\#^F$	$\#^A$	γ^F	γ^A	γ^{MXGA}
			Mean	Median	Max	Mean	Median	Max					
A	1	25	0.0	0.0	0.6	18.4	13.0	69.4	0	0	28.4	13.9	**12.4**
	2	12	11.5	8.8	35.6	20.7	14.3	69.0	0	0	12.8	**2.3**	11.1
	3	24	2.6	1.7	11.6	25.1	18.6	78.0	0	0	46.7	22.1	**22.0**
	4	14	34.0	29.1	87.6	60.4	52.5	197.0	0	8	36.8	**10.5**	17.1
	5	29	2.2	1.7	12.0	26.0	17.9	126.2	0	1	37.9	19.1	**17.9**
	6	24	46.1	42.7	136.2	76.2	58.7	228.8	2	14	67.0	**9.3**	16.6
	7	33	0.8	0.4	7.0	24.6	20.5	82.6	0	0	43.0	27.7	**23.5**
	8	40	1.1	0.4	7.4	27.3	18.3	105.0	0	0	42.2	26.5	**23.3**
	9	2	0.1	0.0	2.0	12.7	0.1	79.8	0	0	3.5	**1.7**	1.7
	10	19	13.3	7.9	76.2	26.0	17.8	108.4	0	0	34.3	**20.2**	23.8
All			11.2	1.2	136.2	31.7	19.6	228.8	2	23			
B	1	23	0.0	0.0	0.0	15.7	10.0	73.6	0	0	47.5	**23.1**	24.2
	2	3	10.3	7.6	50.0	19.1	13.5	74.4	0	0	9.0	**2.9**	34.0
	3	20	2.5	1.4	12.2	17.5	11.7	78.0	0	0	77.8	**41.7**	46.2
	4	21	28.1	20.4	82.0	50.8	35.4	168.8	0	7	40.9	**14.3**	36.0
	5	32	2.2	1.0	14.6	17.8	9.2	154.4	0	1	61.5	**33.3**	35.5
	6	22	39.1	35.4	134.4	63.4	53.3	292.4	2	10	74.6	**12.5**	37.7
	7	46	0.8	0.2	4.8	21.0	9.8	81.8	0	0	64.2	**48.7**	52.2
	8	47	0.7	0.4	3.8	21.1	10.9	107.0	0	0	64.2	**46.5**	49.4
	9	2	0.1	0.0	1.2	8.1	0.0	61.8	0	0	6.2	**2.4**	2.4
	10	21	11.1	6.7	45.6	21.0	14.8	66.4	0	0	57.4	**40.9**	53.5
All			9.5	0.8	134.4	25.5	11.5	292.4	2	18			
C	1	24	0.0	0.0	0.4	11.5	7.9	44.0	0	0	117.9	**69.6**	93.0
	2	14	10.9	9.7	40.2	18.9	15.1	62.0	0	0	23.1	**7.2**	149.5
	3	24	2.3	1.7	7.2	20.2	13.5	75.8	0	0	152.0	**91.2**	124.9
	4	20	33.1	25.5	88.8	57.6	50.4	217.0	0	8	140.0	**35.6**	153.2
	5	28	3.0	2.0	12.0	23.5	12.8	85.0	0	0	146.7	**81.1**	105.0
	6	29	44.6	45.3	139.0	77.8	68.1	278.4	2	12	248.0	**76.9**	241.2
	7	41	1.2	0.6	7.4	25.3	12.8	108.8	0	0	186.3	**156.7**	209.6
	8	37	1.8	0.8	11.0	26.7	17.1	149.4	0	1	301.5	**232.2**	273.3
	9	2	0.2	0.0	1.8	10.4	0.2	66.8	0	0	15.7	10.5	**9.9**
	10	18	15.0	11.1	47.8	30.1	21.2	156.6	0	1	282.9	**214.5**	318.5
All			11.2	1.6	139.0	30.2	16.1	278.4	2	22			
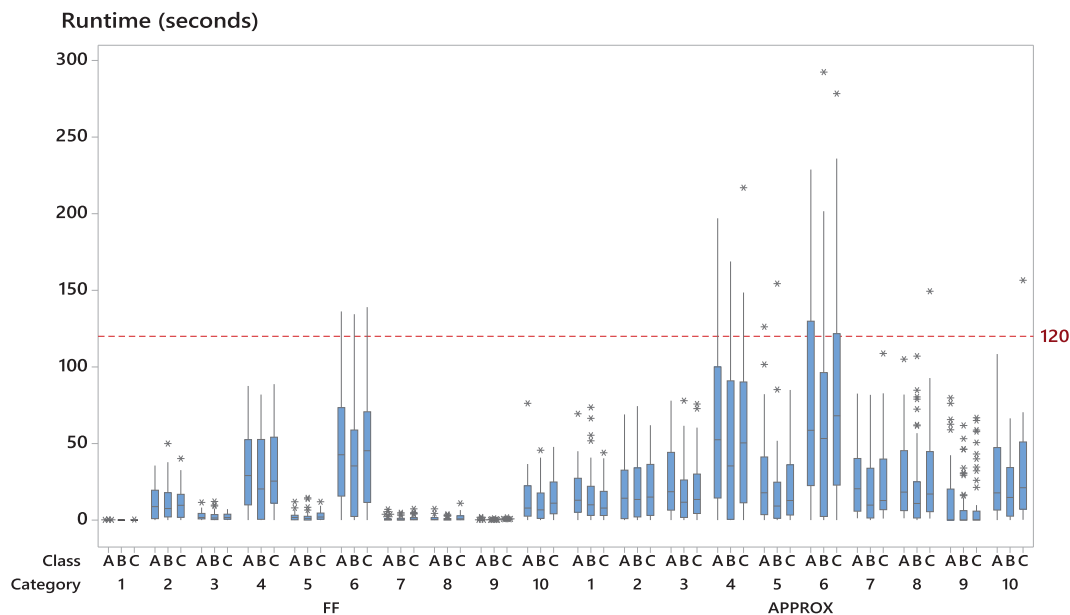
Fig. 4. Box plot for the run times of *FF* and *APPROX* by category and class.

Fig. 6, the mean optimality gap of *APPROX* is smaller than its *MXGA* counterpart, for both sets $\mathcal{L}$ and $\mathcal{S}$ and for all classes. Indeed, the mean difference depends on the class with −1.61, −10.48, and −70.30% point estimates for classes A, B, and C, respectively. These results are most likely due to the nature of *MXGA*. In fact, *MXGA* along with its packing algorithm are strongly oriented towards obtaining a dense packing; a rather important criterion for class A whose instances are characterized by narrow intervals of due dates. However, *MXGA* is myopic when the due dates are sparse. This myopic nature is further highlighted by Fig. 7, which shows that *APPROX* produces better results than *MXGA* more frequently, in particular for class C whose items have a wider range of due dates.
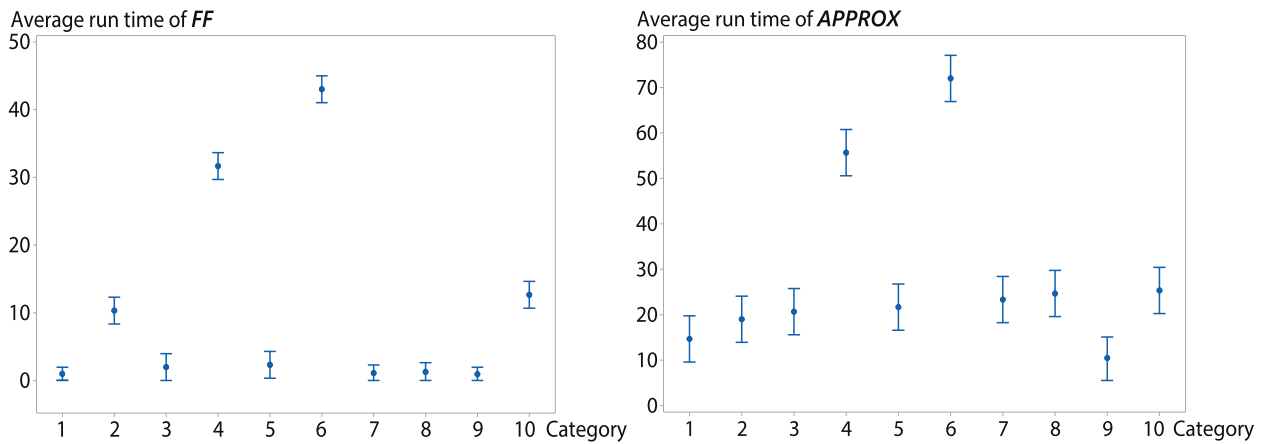
Fig. 5. 95% confidence intervals for the mean run times of *FF* and *APPROX* by category.
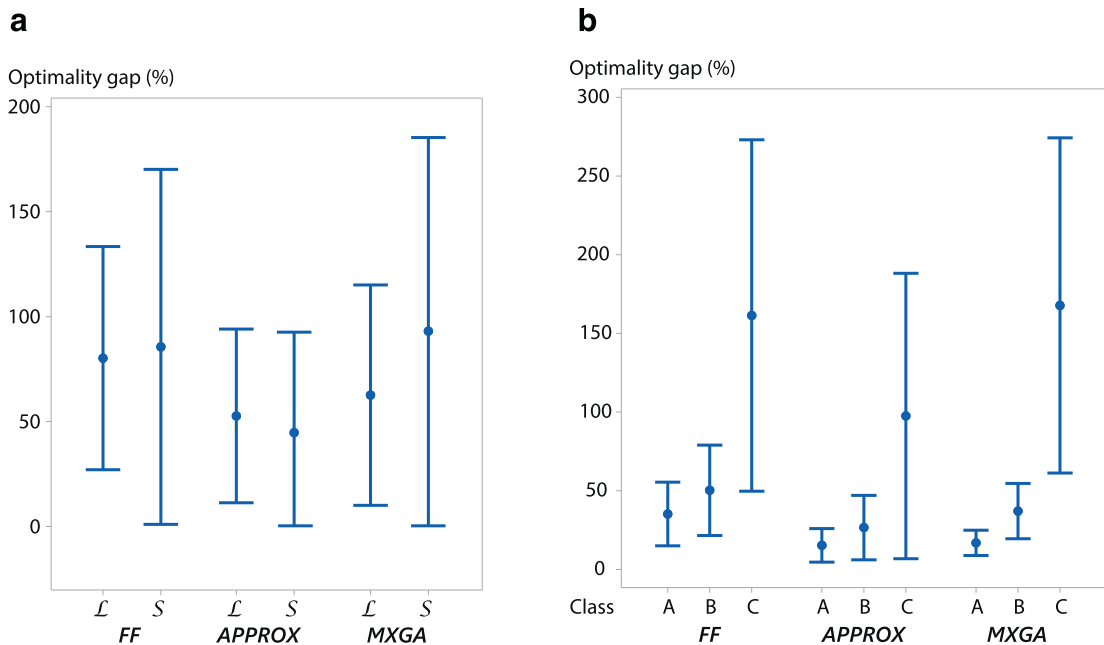

Fig. 6. 95% confidence interval of the mean optimality gaps obtained by *FF*, *APPROX* and *MXGA* (a) by set (b) and by class.

The size of the optimality gaps seems unusually high. Even though part of it may be due to the quality of the upper bounds obtained by *FF*, *APPROX* and *MXGA*, most of it is most likely due to the looseness of the lower bound. The comparison of Tables 4 and 5 further supports this claim. Tables 4 and 5 report the percent deviations of *FF* and *APPROX* from the best lower bound LB^* and LB_1, respectively. For *FF*, this gap is 282.9% for class C, category 10 in Table 5, but reduces to 204% in Table 4. A good example is rand10.txt_C_1, the 10th instance of category 1, class C, $n = 20$. The percent gaps of *FF* and *APPROX* are 136% when computed with respect to $LB_1 = 69$, but become 0% when computed with respect to $LB_3 = 163$. In this case, the upper bound matches LB_3, and proves the optimality of the solutions obtained by *FF* and *APPROX*. Even though the instances are numerous, we only cite a second example: rand6.txt_C_10 of category 10, class C, $n = 20$, where $LB_1 = 3$ results in an optimality gap of 1533% whereas $LB_3 = 92$ proves the optimality of the upper bound obtained by *APPROX*.

In summary, the proposed approach enhances many existing upper bounds, assesses the tightness of existing and proposed lower bounds, and proves the optimality of many open benchmark problems. It outperforms *MXGA* in terms of solution quality, run time, and number of proven optima.

6.5. Performance of FF and APPROX on large-sized instances

This section investigates the performance of *FF* and *APPROX* for very large instances. For this purpose, it expands the current benchmark set of $n = 100$ to a set with $n = 100\tau$, $\tau = 2, \ldots, 5$, by duplicating the items of each instance τ times. It creates 10 new instances per class, category and problem size; generating the due dates for the new items as Section 6.1 explains.

For large-sized instances, *FF* may require a sizeable number of iterations to reach a feasible solution. A large ratio of number of items per bin makes the lower bound in *FF* loose; thus ineffective for the packing test. As a result, *PACK* is called numerous times with most of these calls being useless due to their large problem size. *FF* checks feasibility after appending a single item. Every feasibility check increases the runtime. A number of heuristic strategies can be designed to limit such ineffective calls. *Two such strategies follow.*

The *simplest* strategy limits the search of Loop (12–16) of Algorithm 1 to σ items among the items that succeed item i in N. This involves a new counter, which is incremented every time Condition (13) fails, and is re-initiated to zero otherwise. The loop stops when the counter equals σ.
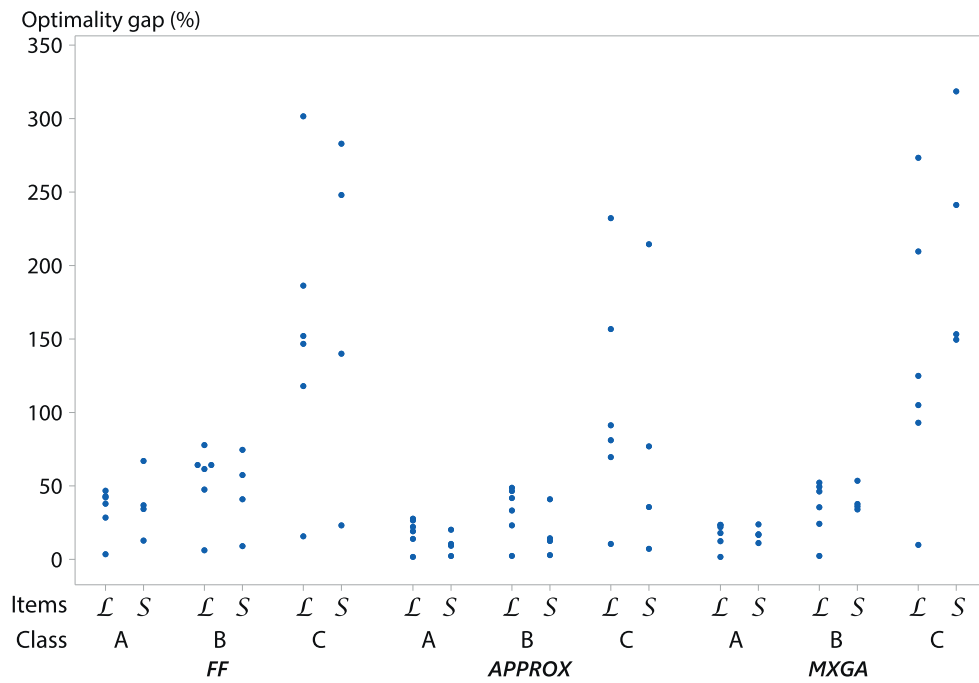
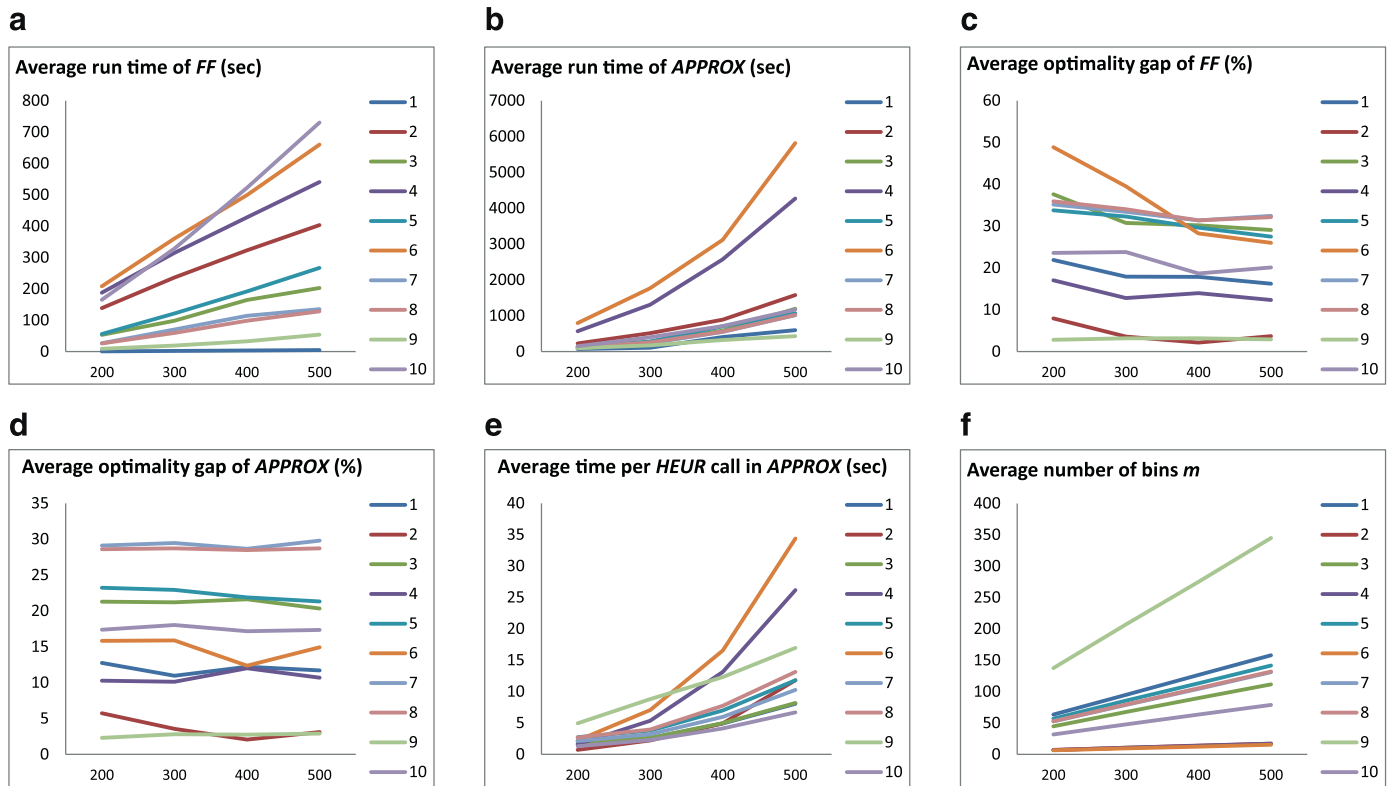**Fig. 7.** Observed percent optimality gaps of *FF*, *APPROX* and *MXGA* by class.

Fig. 8. Summary results for large-sized instances of class A.

A *second* strategy monitors the maximal dimension μ of an item that can be packed in the current bin k. Initially, $\mu = max\{W, H\}$. When item i violates Condition (13), a supplementary test $PACK(N_k \cup \{i'\})$ is run to check the feasibility of packing a dummy item i' defined by $w_{i'} = max\{w_i, h_i\}$ and $h_{i'} = 1$ into k. If the test reveals the infeasibility of such a packing, μ is updated to $w_{i'}$, and Loop (12–16) skips any subsequent item i whose $max\{w_i, h_i\} \geq \mu$. The additional calls to *PACK* increase the runtime of the heuristic; a negligible increase compared to the runtime saved by the omitted unsuccessful future calls to *PACK*. That is, this strategy is efficient only when n is large. Because *PACK* is a heuristic, its outcome may suggest an infeasible packing while a feasible one exists. In fact, the choice of μ is not based on an exact method but on a rule of thumb. Thus, the application of *PACK* may slightly degrade the solution quality.

Similarly, *APPROX* may require too many iterations of *HEUR* (and *HEUR'*), with each resulting in a slight improvement of *UB*. To avoid a series of too many small enhancements, *APPROX* adopts the minimal improvement technique, which first computes a value for *UB* using *FF*, and iteratively examines feasible solutions whose lateness is $\delta\%$ smaller than the current *UB*. When it fails to find a solution whose lateness is $\delta\%$ better than the incumbent, it examines solutions sequentially (as Section 5.3 explains) until no further improvement can be achieved.

Herein, *FF* and *APPROX* are run with a 3-second time limit for *PACK*, and maximal *count* number of iterations $a_{lim}^{HEUR}=30$ and $a_{lim}^{HEUR'}=10$. In addition, the two heuristic strategies are applied for instances of set $\mathcal{S}$ with $\sigma=40$. Despite their slightly weaker solutions' quality, the two strategies reduce the runtime by an order of magnitude for instances with 500 items. Finally, the minimal improvement technique is used with $\delta=2\%$. It speeds significantly the search for a local optimum for instances with $n>100$. The summary of the results for class A is displayed in Fig. 8. The results for the other two classes behave similarly, and are therefore omitted from further consideration. Fig. 8(a)–(d) depicts the average run times for *FF* and *APPROX* along their optimality gaps, computed with respect to LB_1. Fig. 8(e) illustrates the average runtime per *HEUR* call in *APPROX*. This runtime sums the times required by the iterative calls to *ASSIGN* until either a feasible solution is reached or infeasibility is detected. Finally, Fig. 8(f) reports the average number of bins used per problem size and category.

Fig. 8(a) shows the linear growth of the average runtime of *FF* as a function of the number of items with a near to one correlation of determination. This trend is in turn dictated by the perfectly linear growth of the number of bins as n increases, as Fig. 8(f) illustrates. This is expected as the time limit for *PACK* is a prefixed constant. To the opposite and as Fig. 8(b) suggests, the average runtime of *APPROX* is not linearly proportional to n as there is no time limit on MIP. Categories 4 and 6 have the hardest instances. Their mean run times differ from those of all other categories. These two categories have the largest number of items per bin ratio as well as the largest variable domains that are used by CP and MIP. Subsequently, they both require a larger mean runtime per *HEUR* call as Fig. 8(e) shows, and a larger number of *HEUR* calls while the *ASSIGN* sub-problem itself is harder. The mean run times of both *FF* and *HEUR* are category, set and size dependent.

The mean optimality gap of *FF* is category dependent. It is not however set or size dependent. On the other hand, the mean optimality gap of *HEUR* is both category and set dependent but is not size dependent. *HEUR* reduces the optimality gap of all the solutions obtained by *FF*. This can be further inferred by the comparison of Fig. 8(c) and (d). The mean reduction is 4.61%. It is neither set nor size dependent but is category dependent. It is largest for category 6 with point and 95% confidence interval estimates of 9.61% and (17.21%, 24.64%), respectively.

7. Conclusion

This paper addresses the two-dimensional non-oriented bin packing problem with due dates. It proposes a lower bound, an exact mixed integer model, and an approximate approach that significantly enhances existing results on many benchmark instances from the literature. It solves 33.93% of the instances to optimality. Because the exact model can be solved to optimality by an off-the-shelf solver, the total percent of instances solved to optimality is 39.07%. Unlike many traditional constructive packing heuristics, the packing approach packs simultaneously several items into several bins and takes advantage of the feasibility constraints to guide the search to a local optimum. Its concept of free regions is not specific to the due dates complicating constraints. It makes the proposed approach easily adaptable to other complex bin-packing related problems with problem-specific constraints such as routing, time windows, and tardiness related costs. Since the dual feasible functions are applicable to higher-dimensional packing, the approach can also be extended to this area.

References

Alves C., Clautiaux F., de Carvalho J. V., Rietz J.. Dual-feasible functions for integer programming and combinatorial optimization: Basics, extensions and applications. EURO advanced tutorials on operational research. 2016. Springer International Publishing. 10.1007/978-3-319-27604-5-4.

Arbib, C., & Marinelli, F. (2014). On cutting stock with due dates. *Omega, 46,* 11–20. doi:10.1016/j.omega.2014.01.004.

Arbib, C., & Marinelli, F. (2017). Maximum lateness minimization in one-dimensional bin packing. *Omega, 68,* 76–84. doi:10.1016/j.omega.2016.06.003.

Bennell, J. A., Lee, L. S., & Potts, C. N. (2013). A genetic algorithm for two-dimensional bin packing with due dates. *International Journal of Production Economics, 145,* 547–560.

Burke, E., Hyde, M., & Kendall, G. (2006). Evolving bin packing heuristics with genetic programming. In *Proceedings of the 2006 parallel problem solving from nature – PPSN IX: 4193* (pp. 860–869). Springer. Lecture notes in computer science doi: 10.1007/11844297-87.

Chen, C., Lee, S., & Shen, Q. (1995). An analytical model for the container loading problem. *European Journal of Operational Research, 80,* 68–76.

Clautiaux, F., Jouglet, A., Carlier, J., & Moukrim, A. (2008). A new constraint programming approach for the orthogonal packing problem. *Computers & Operations Research, 35,* 944–959. doi:10.1016/j.cor.2006.05.012.

Clautiaux, F., Jouglet, A., & Hayek, J. E. (2007). A new lower bound for the non-oriented two-dimensional bin-packing problem. *Operations Research Letters, 35,* 365–373. https://doi.org/10.1016/j.orl.2006.07.001

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

Fekete, S. P., & Schepers, J. (2004). A general framework for bounds for higher-dimensional orthogonal packing problems.. *Mathematical Methods of Operations Research, 60,* 311–329.

Belov, G., Mukhacheva, E. A., & Scheithauer, G. (2008). One-dimensional heuristics adapted for two-dimensional rectangular strip packing. *The Journal of the Operational Research Society, 59,* 823–832.

Iori, M., & Martello, S. (2013). An annotated bibliography of combined routing and loading problems. *Yugoslav Journal of Operations Research, 23,* 311–326. doi:10.2298/YJOR130315032I.

Jansen, K., & Pradel, L. (2016). New approximability results for two-dimensional bin packing. *Algorithmica, 74,* 208–269.

Johnson, D. S., Demers, A., Ullman, J., Garey, M., & Graham, R. (1974). Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing, 3,* 299–325.

Laborie, P. (May 27-31, 2009). IBM ILOG CP Optimizer for detailed scheduling illustrated on three problems. In W. van Hoeve, & J. N. Hooker (Eds.), *Proceedings of the sixth international conference on integration of AI and OR techniques in constraint programming for combinatorial optimization problems, (CPAIOR 2009)* (pp. 148–162). Pittsburgh, PA, USA: Springer. doi:10.1007/978-3-642-01929-6-12.

Laborie, P., & Rogerie, J. (May 15-17, 2008). Reasoning with conditional time-intervals. In *Proceedings of the twenty-first international florida artificial intelligence research society conference* (pp. 555–560). Coconut Grove, Florida, USA.

Li, S. (1996). Multi-job cutting stock problem with due dates and release dates. *The Journal of the Operational Research Society, 47,* 490–510. doi:10.2307/3010726.

Lodi, A., Martello, S., Monaci, M., & Vigo, D. (2014). Two-dimensional bin packing problems. In V. T. Paschos (Ed.), *Paradigms of combinatorial optimization: Problems and new approaches* (pp. 107–129). Hoboken, NJ, USA: Wiley-ISTE. doi:10.1002/9781119005353.ch5. 2nd ed.

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

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

Onodera, H., Taniguchi, Y., & Tamaru, K. (1991). Branch-and-bound placement for building block layout. In *Proceedings of the twenty-eighth ACM/IEEE design automation conference (DAC '91)* (pp. 433–439). New York, NY, USA: ACM.

Polyakovskiy, S., Makarowsky, A., & M'Hallah, R. (2017). Just-in-time batch scheduling problem with two-dimensional bin packing constraints. In *Proceedings of the genetic and evolutionary computation conference (GECCO '17)* (pp. 321–328). New York, NY, USA: ACM. doi:10.1145/3071178.3071223.

Polyakovskiy, S., & M'Hallah, R. (2011). An intelligent framework to online bin packing in a just-in-time environment. In *Modern approaches in applied intelligence: 6704* (pp. 226–236). Springer. doi:10.1007/978-3-642-21827-9-23.

Reinertsen, H., & Vossen, T. W. (2010). The one-dimensional cutting stock problem with due dates. *European Journal of Operational Research, 201,* 701–711. doi:10.1016/j.ejor.2009.03.042.

Sim, K., & Hart, E. (2013). Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model. In *Proceedings of the genetic and evolutionary computation conference, GECCO '13* (pp. 1549–1556). doi:10.1145/2463372.2463555.

Sim, K., Hart, E., & Paechter, B. (2015). A lifelong learning hyper-heuristic method for bin packing. *Evolutionary Computation, 23*, 37–67. doi:10.1162/EVCO-a-00121.

Verstichel, J., Kinable, J., De Causmaecker, P., & Vanden Berghe, G. (2015). A combinatorial benders' decomposition for the lock scheduling problem. *Computers & Operations Research, 54*, 117–128. doi:10.1016/j.cor.2014.09.007.

Vilím, P. (September 20-24, 2009). Edge finding filtering algorithm for discrete cumulative resources in $\mathcal{O}(kn \log n)$. In I. P. Gent (Ed.), *Proceedings of the fifteenth international conference on principles and practice of constraint programming – CP 2009 (Lisbon, Portugal)* (pp. 802–816). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-04244-7-62.

Vilím, P., Laborie, P., & Shaw, P. (2015). Failure-directed search for constraint-based scheduling. In L. Michel (Ed.), *Proceedings of the integration of AI and or techniques in constraint programming*. In *Springer international publishing: 9075* (pp. 437–453). doi:10.1007/978-3-319-18008-3-30.