Number of bins and maximum lateness minimization in two-dimensional bin packing

Number of bins and maximum lateness minimization in two-dimensional bin packing

Claudio Arbib[1]

Università degli Studi dell'Aquila
Dipartimento di Ingegneria/Scienze dell'Informazione e Matematica
and
Centre of Excellence DEWS
via Vetoio, I-67010 L'Aquila, Italy

Fabrizio Marinelli[2], Andrea Pizzuti[3]

Università Politecnica delle Marche
Dipartimento di Ingegneria dell'Informazione
via Brecce Bianche, I-60131 Ancona, Italy

Abstract

In this work we address an orthogonal non-oriented two-dimensional bin packing problem where items are associated with due-dates. Two objectives are considered: minimize (i) the number of bins and (ii) the maximum lateness of the items. We discuss basic properties of non-dominated solutions and propose a sequential value correction heuristic that outperforms two benchmark algorithms specifically designed for this problem. We also extend the benchmark dataset for this problem with new and larger industrial instances obtained from a major manufacturer of cutting machines. Finally, we give some insights into the structure of Pareto-optimal sets in the classes of instances here considered.

Keywords: Bin packing, Scheduling, Heuristics, Multi-objective Problems.

(Work supported by the Italian Ministry of Education and Research, National Research Program PRIN 2015, contract n. 20153TXRX9)

1. Introduction

The r-dimensional Bin Packing (rBP) is a well-known combinatorial

[1]claudio.arbib@univaq.it
[2]fabrizio.marinelli@staff.univpm.it
[3]a.pizzuti@pm.univpm.it

optimization problem [18] that calls for packing a given set $\mathcal{I}$ of m r-dimensional items into a minimum number of identical r-dimensional bins. An instance of the two-dimensional basic version (2BP) consists of m rectangular items described by positive integer widths $w_1, \ldots, w_m$ and heights $h_1, \ldots, h_m$ and (a sufficient number of) rectangular bins of size $W \times H$, with $W \geq w_i$ and $H \geq h_i$, $i = 1, \ldots, m$. In general, a packing is feasible if items are completely contained in the bins and do not overlap each other, but a number of further placement rules and restrictions have been investigated in the last years (see e.g. [27] for a comprehensive survey). In this work we deal with the *orthogonal* 2BP, where item edges must be parallel to bin edges. Both the *oriented* and the *non-oriented* versions are considered: in the latter case, 90° item rotation is allowed. Following a common use, we will denote these problems as 2OBP, 2RBP, where O, R stand for *oriented* and *rotation*, respectively.

A 2BP solution that is optimal with respect to the efficiency of bin allocation, may not be so attractive when its quality depends also on the way items are allocated over time: in many application, the bins of a 2BP solution are in fact packed in some sequence, and since filling each individual bin requires a certain amount of time, this sequence will complete the packing of each item (making it available, e.g., for delivery) in some specific time: if item i is due by time d_i, one can then be interested in optimizing a due-date related objective function, such as weighted tardiness, number of tardy jobs, maximum lateness, etc. This idea refers BP to a special case of what is commonly known as Single Batch-processing Machine Scheduling problem: jobs (items) are done in batches on a machine, a batch (bin) has a limited capacity and one wants to minimize some function of job completion times. An application of the above scheduling problem can be found in semiconductor manufacturing, where burn-in operations on silicon plates containing batches of chips must be scheduled, see e.g. [26].

The mainstream applications of 2BP are truck loading and warehousing in logistics, and cutting optimization in manufacturing. In these areas, the interest for scheduling derives from multiple aspects, first of all the fulfillment of order delivery dates, for which the optimization of operation schedule can be worth of consideration. Other applications of 2OBP can be found in telecommunications: in [16] the authors model the downlink subframe allocation problem in Mobile WiMAX technology as a 2OBP where

items are group of data packets and bins represent time/frequency slots. A related packing-scheduling problem is addressed in [11], referred to QoS in a UMTS system with data packets obeying to deadlines or minimizing delays.

Due to the illustrated significance, scheduling issues in packing problems are receiving increasing attention. For instance: [2] investigates the minimization of a convex combination of number of bins and maximum lateness; [3] and [23] propose Integer Linear Programming formulations to minimize the number of bins *and* the weighted tardiness in one-dimensional cutting stock. For a comprehensive view of the state-of-the-art, we refer to exhaustive literature reviews reported in [6, 21].

In this paper we consider a bi-criteria 2BP with due dates: we refer to this problem as 2BP-DD in general, and as 2OBP-DD, 2RBP-DD when orientation is significant. The paper develops earlier ideas formulated in [19]: problems are described in Section 2 with a discussion of some meaningful properties; a Sequential Value Correction heuristic (SVC-DD) for the problems addressed is proposed in Section 4; an improved dual bound for the scheduling objective function is provided in Section 5; finally, in Section 6, non-dominated solutions computed by the SVC-DD heuristic are analysed and compared to those obtained by the algorithms proposed in [6] and [21].

2. Problem and basic properties

In 2BP-DD, we assume as in [6] that

Assumption 1. *The bin processing time $\tau \in \mathbb{Z}_+$, i.e., the time required to fill any bin with the assigned items, is known, constant and independent on both the items packed and the shape of the packing pattern.*

Assumption 2. *The completion time of any item i contained in the k-th bin of the sequence is $C_i = k\tau$.*

Rephrasing the above assumptions, bins are completed and delivered at a constant rate with intervals not smaller than the actual time required by the packing operations (Assumption 1), and all the items of a bin are simultaneously released as soon as the bin is completed (Assumption 2).

Our objective calls for the minimization of both the number N of bins used to pack all items, and the maximum lateness $L_{\max}$ of an item. By Assumption 2, the number of bins used is proportional to $\frac{1}{\tau}$ and to the largest completion time $C_{\max} = \max_{i \in \mathcal{I}}\{C_i\}$ of the items in $\mathcal{I}$. The lateness

$L_{\max}$ is defined in the literature as the largest violation of a due date, and can be a negative number. For our purposes, however, we regard negative $L_{\max}$ as equivalent to $L_{\max} = 0$, and so define $L_{\max} = \max_{i \in \mathcal{I}}\{L_i, 0\}$, where $L_i = C_i - d_i$ is the lateness of item i, and d_i is its due-date.

Once sequenced, a feasible packing has a value defined by a pair $\pi = (n, \ell)$ with $N = n$ and $L_{\max} = \ell$. This packing is said to be (strictly) *non-dominated* if for any feasible packing of value $(\bar{n}, \bar{\ell})$ one either has $n < \bar{n}$ or $\ell < \bar{\ell}$ (or both). Moreover, a pair (n, ℓ) is said to be *weakly non-dominated* if there is no feasible pair $(\bar{n}, \bar{\ell})$ such that $\bar{n} < n$ and $\bar{\ell} < \ell$. For example, take $\pi_1 = (20, 5), \pi_2 = (18, 6), \pi_3 = (18, 5), \pi_4 = (21, 6)$: in this set π_1, π_2, π_3 are all weakly non-dominated, π_3 is strictly non-dominated and weakly dominates π_1, π_2 and finally π_4 is strictly dominated by π_1, π_3 and weakly dominated by π_2. The general problem (viz., regardless of item orientation) can then be formalized in this way:

Problem 1. *Given a set $\mathcal{I}$ of m items of size $w_i \leq W, h_i \leq H$, each due by a specific date d_i, $i \in \mathcal{I}$, find all the item allocations to bins whose values $(N, L_{\max})$ are strictly non-dominated.*

Let us now focus on some properties of non-dominated solutions of Problem 1: specifically, on the dependence of $L_{\max}$ on the filling time τ (§2.1), and on upper bounds that can be obtained exploiting the relation between N and $L_{\max}$ (§2.2).

2.1. Dependence on τ

Non-dominated solutions of 2BP-DD depend on τ in a non-trivial way. Suppose to draw the Pareto region for given due-dates d_i and variable τ. As a general observation, the smaller the τ, the smaller the minimum $L_{\max}$, while the minimum N does not depend on τ and is therefore unchanged. But, contrary to intuition, the relative positions of Pareto-optima are not necessarily preserved as τ decreases. In fact, let $\mathcal{I}$ be a 2BP-DD instance with bin processing time $\tau > 1$ and due dates $d_i, i \in \mathcal{I}$. Let then x^1 and x^2 be two solutions of $\mathcal{I}$ with $N = n$ bins and minimum $L_{\max}$ of

$$\ell^1 = k\tau - d_p < h\tau - d_q = \ell^2$$

for some k, h, p and q: that is, p and q are the *critical items* determining the minimum $L_{\max}$ in the respective solutions, p (q) is processed in the k-th (in the h-th) bin, and x^1 weakly dominates x^2. Suppose that p and q

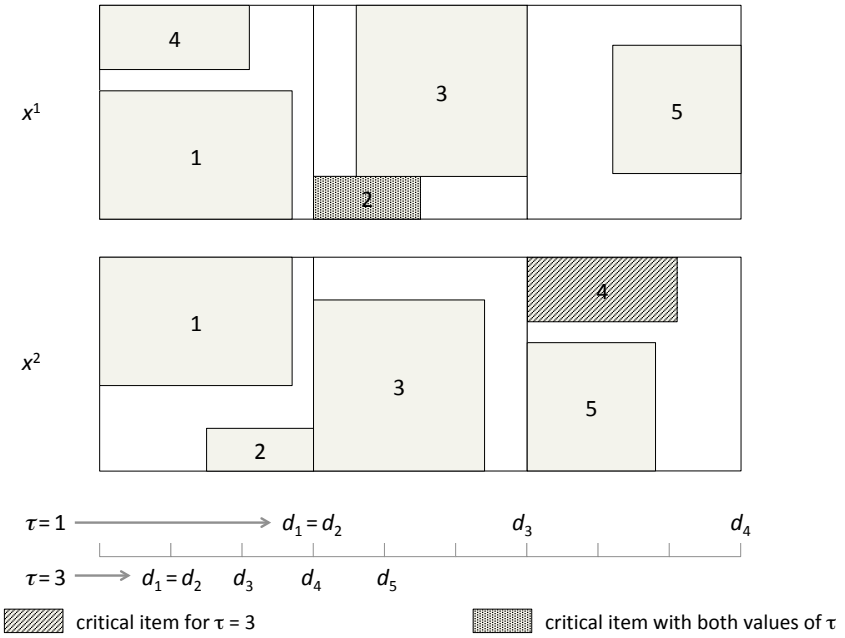

Figure 1: How the Pareto region changes with τ, *ceteris paribus*.

still determine the minimum $L_{\max}$ in x^1 and x^2 when bin processing time is altered to $\tau' = \alpha\tau$. It is easily seen that x^2 now weakly dominates x^1 for α such that

$$(\alpha - 1)\tau(h - k) < \ell^1 - \ell^2 \tag{1}$$

More explicitly (see Figure 1), take an instance with $m = 5$ items of size $(w_1, h_1) = (9, 6)$, $(w_2, h_2) = (5, 2)$, $(w_3, h_3) = (8, 8)$, $(w_4, h_4) = (7, 3)$, $(w_5, h_5) = (6, 6)$ and due-dates $d_1 = d_2 = 1$, $d_3 = 2$, $d_4 = 3$ and $d_5 = 4$. Suppose $\tau = 3$ and $W = H = 10$. Consider two solutions x^1 and x^2, both fitting in $N = 3$ bins and sequenced in this way: x^1 has items 1 and 4 packed in the first bin, items 2 and 3 in the second, and item 5 in the third; in x^2, instead, the first bin is filled with items 1 and 2, item 3 is packed in the second, and the remaining items in the third. Then, x^1 weakly dominates x^2 since $\ell^1 = 2\tau - d_2 = 5$ and $\ell^2 = 3\tau - d_4 = 6$. However x^2 weakly dominates x^1 for $0 < \alpha \leq \frac{2}{3}$: e.g., for $\alpha = \frac{1}{3}$, the minimum $L_{\max}$ become $\ell^1 = 1$ and $\ell^2 = 0$.

Due-date scaling by τ is however an easier matter. In fact, an instance $\mathcal{I}$ with $\tau > 1$ is clearly equivalent to $\mathcal{I}'$ with $\tau' = 1$ and fractional due-dates

d_i/τ. When due-dates are restricted to be integers, the above transformation is of course not allowed but, as we will immediately show, the dependence on τ can be controlled and turns out to be not excessive. Construct in fact a scaled instance $\mathcal{I}'$ with due dates

$$d'_i = d_i - (d_i \mod \tau)$$

rounded down to the closest multiple of τ. Take two non-dominated solutions x of $\mathcal{I}$ and x' of $\mathcal{I}'$ with n bins and minimum $L_{\max}$ of values ℓ, ℓ', respectively. Then

Proposition 1. $\ell' < \ell + \tau$.

Proof. Let C_i, L_i and C'_i, L'_i be the completion time and the lateness of item i in solutions x, x'. Therefore

$$C_i - d'_i = L_i + (d_i \mod \tau)$$

is the lateness of item $i \in I$ in solution x re-computed according to the scaled due dates. Because x' is $L_{\max}$-optimal in the scaled problem,

$$\ell' = \max_i\{C'_i - d'_i\} \leq \max_i\{C_i - d'_i\} \leq \max_i\{L_i + (d_i \mod \tau)\} \leq$$

$$\leq \ell + \max_i\{(d_i \mod \tau)\} < \ell + \tau$$

□

Proposition 1 can easily be extended to $1 < \tau' < \tau$, obtaining in general $\ell' < \ell + \frac{\tau}{\tau'}$. It basically states that the increase of $L_{\max}$ is strictly limited to the time required to pack a single bin. Moreover, it is easy to see that the frontier of the strictly non-dominated solutions of $\mathcal{I}$ can be obtained by the set of all the weakly non-dominated solutions of $\mathcal{I}'$.

2.2. Upper bounds to N

Clearly, minimizing N does not correspond to minimizing $L_{\max}$: Example 2.3 in [2] shows that a solution with a minimum number N^* of bins and $L_{\max} = N^* - 1$ can be converted into one with $N^* + 1$ bins and $L_{\max} = 0$. However, a strong correlation between the two objectives exists due to Assumption 2. Indeed, given a solution of value (n, ℓ), let

$$I_\ell = \{i \in \mathcal{I} : L_i < \ell\}$$

denote the set of items delayed less than ℓ. For any $i \in I_\ell$, define

$$\Delta_i(\ell) = \left\lfloor \frac{\ell - L_i}{\tau} \right\rfloor$$

that is, $\tau\Delta_i(\ell)$ is the largest delay of item i that does not affect the lateness bound ℓ: in symbols, $L_i + \tau \cdot \Delta_i(\ell) \leq \ell \ \forall i \in I_\ell$. Let C_i, an integer multiple of τ, denote the completion time of $i \in \mathcal{I}$ in a solution of value (n, ℓ). As already observed in [21],

Proposition 2. *For any solution of 2BP-DD with lateness $L_{\max} = \ell$,*

$$N \ \leq \ \max_{i \in I_\ell} \{C_i/\tau + \Delta_i(\ell)\}$$

Proof. By the above definition, in any solution with lateness ℓ, item i is delayed at most $\tau\Delta_i(\ell)$. So its completion time cannot be more than $C_i + \tau\Delta_i(\ell)$, and therefore the number N of bins of any such solution is no larger than indicated. □

Solutions of the single-objective 2BP can be exploited to obtain bounds to non-dominated solutions of 2BP-DD. Let items be sorted by early due date first (EDD), that is, $d_1 \leq \ldots \leq d_n$, and let $(n_{EDD}, \bar{\ell})$ be the value of a 2BP-DD solution computed by a Next Fit algorithm regardless of lateness optimization. Then:

Proposition 3. *Any non-dominated solution has $N \leq n_{EDD}$ bins.*

Proof. Let $J = \{i \in \mathcal{I} : \frac{C_i}{\tau} + \Delta_i(\bar{\ell}) \geq n_{EDD} + 1\}$. Due to the Next Fit rule, if the item in J with the smallest due-date is packed in the k-th bin, then bins from $k+1$ to n_{EDD} contain only items of J. The statement is trivial for $J = \emptyset$. Suppose $J \neq \emptyset$. If there exists a permutation of the items in the first k bins with $L_{\max} < \bar{\ell}$, then an additional bin is useless. Otherwise, let $h \in \mathcal{I}$ be an item defining $\bar{\ell}$ (namely, $L_h = \bar{\ell}$). A better solution can only be achieved by exchanging h with another item i packed in one of the previous bins of the current solution. However, $d_i \leq d_h$ holds for all these items and the exchange would imply $L_{\max} \geq \bar{\ell}$. □

In other words, Proposition 3 says that N and $L_{\max}$ trade off only for N up to a given threshold, beyond which $L_{\max}$ increases with N.

3. Approximating non-dominated solutions

In multicriteria optimization, several methods were defined to exactly identify the set of non-dominated solutions. Generally speaking, such methods can be distinguished in scalarization-based (e.g., weighted combination, ϵ-constraint, Tchebycheff metric) and non-scalarizing (e.g., lexicographic, max-ordering), see [25]. Whatever method is adopted, providing a complete description of the Pareto set is computationally challenging, and an approximated description of the Pareto set via (meta)heuristics is often an acceptable compromise [13]. Nevertheless, any approximation raises the question of how to evaluate the quality of the frontier found, and which approximation measures are the most adequate. In the literature, several measures were proposed [28], but just few of them consider worst case analysis.

To the best of our knowledge, there are two main approaches to measure worst case approximation of algorithms for p-criteria problems: one [12], discussed in §3.1, is suitable for heuristics that produce a single solution to be compared to the whole Pareto frontier, the other [20] is used for algorithms that produce a set of solutions that do not dominate each other. In particular, according to [20], an algorithm $\mathcal{H}$ is ϵ-approximated if, for every non-dominated solution x^*, there exists a solution $x_{\mathcal{H}}$ such that

$$f_j(x_{\mathcal{H}}) \leq (1+\epsilon) f_j(x^*)$$

holds for each criterion j. However, this kind of approximation cannot be adopted in our case (and in general when the optimal value of one of the criteria can be zero) because the ratio $L_{\max}(x_{\mathcal{H}})/L_{\max}(x^*)$ is not well defined.

If instead we consider the absolute error $E_A(\mathcal{H}) = L_{\max}(x_{\mathcal{H}}) - L_{\max}(x^*)$, we observe as expected that most ϵ-approximate algorithms $\mathcal{H}_{2BP}$ for 2BP can perform arbitrarily bad, i.e., the bound $E_A(\mathcal{H}_{2BP}) \leq \tau(1+\epsilon)N(x^*) - \tau$ is tight. One exception is the Next Fit algorithm, where items are sorted by EDD, which is 3-approximate for 2BP. Indeed:

Proposition 4. *With the Next Fit heuristic, the absolute error of $L_{\max}$ is bounded by $3\tau N(x^*) - \tau$*

Proof. see [22], Corollary 3. □

3.1. Single solution approximation

According to the framework proposed in [12], the performance of an algorithm $\mathcal{H}$ can be measured by relying on two norm-based definitions of approximation ratio:

$$R_1(x_{\mathcal{H}}, x^*) = \frac{|\|f(x_{\mathcal{H}})\| - \|f(x^*)\||}{\|f(x^*)\|} \quad \text{or} \quad R_2(x_{\mathcal{H}}, x^*) = \frac{\|f(x_{\mathcal{H}}) - f(x^*)\|}{\|f(x^*)\|}$$

where $x_{\mathcal{H}}$ is a solution provided by $\mathcal{H}$, x^* is a non-dominated solution, $f(x) \in \mathbb{R}^p$ denotes the objective function vector and $\|\cdot\|$ is a monotone norm. Note that R_1 compares the norms of the vectors, whereas R_2 evaluates the difference of vector components.

We then say that an algorithm $\mathcal{H}$ is R_i-approximated with $\mu \in \mathbb{R}_+$ if $x_{\mathcal{H}}$ fulfils

$$R_i(x_{\mathcal{H}}, x^*) \leq \mu \tag{2}$$

for all non-dominated solutions x^*. As (2) gives a tighter definition of approximation for $i = 1$ than for $i = 2$, an algorithm $\mathcal{H}$ that is R_1-approximated with some $\mu \in \mathbb{R}_+$ is also R_2-approximated with the same μ. Note that this definition covers two types of approximation, one of which derives from taking a single (heuristic, but even non-dominated) solution to represent the whole Pareto-frontier.

Of course, the approximation ratio depends in general on the norm used. In the following we will refer to the general norm $\|.\|_q$. Observe that for $q = 1$, R_1-approximation reads in our case

$$\frac{N(x_{\mathcal{H}}) + L_{\max}(x_{\mathcal{H}}) - N(x^*) - L_{\max}(x^*)}{N(x^*) + L_{\max}(x^*)} \leq \mu$$

that corresponds to the μ-approximation in the ordinary sense of the single objective problem where one wants to minimize $N + L_{\max}$. As noticed in [2] (Proposition 2.1), this problem can be approximated by heuristics for the traditional 2BP problem. In fact, 2BP can be 1-approximated both in the *oriented* [15] and the *non-oriented* [14] case. Therefore $\frac{n_{2BP} - N^*}{N^*} \leq 1$, where N^* is the minimum feasible number of bins and n_{2BP} is the number of bins used by the approximating algorithm $\mathcal{H}_{2BP}$. For $\tau = 1$ this guarantees an R_1-approximation of Problem 1 with $\mu = 3$ under the $\|.\|_1$ norm, see [2].

The definition of 1-approximation trivially implies $\frac{n_{2BP} - (N^* + k)}{N^* + k} \leq 1$ for

any $k \in \mathbb{N}$. Now note that one can explore all the possible n-values of the Pareto-optimal set by varying k, with at most one non-dominated solution for any given k.

So let $\mathcal{H}_{2BP}$ be a 1-approximate algorithm for 2BP returning a solution of value (n_{2BP}, ℓ_{2BP}), and let $(\bar{n} = N^* + k, \bar{\ell})$ be the value of a non-dominated solution of 2BP-DD, where N^* denotes the minimum number of bins of 2BP and $k \in \mathbb{N}$. For the following two results it is useful to recall that, for $\tau \geq 1$, the maximum lateness of any solution can never exceed its total completion time, i.e., for any 2BP-DD solution of value (n, ℓ) one has $\ell \leq \tau n$; moreover, $\mathcal{H}_{2BP}$ is 1-approximating, hence $n_{2BP} \leq 2N^*$, and therefore $n_{2BP} \leq 2(N^* + k)$ clearly holds for $k \geq 0$ and any solution of $\mathcal{H}_{2BP}$.

Proposition 5. *$\mathcal{H}_{2BP}$ is R_1-approximated for 2BP-DD under $\|.\|_q$ with*

$$\mu = 2\sqrt[q]{1 + \tau^q} - 1$$

for any $\tau \geq 1$.

Proof.

Writing (2) for 2BP-DD and $i = 1$, we obtain

$$(1 - \mu)\sqrt[q]{(N^* + k)^q + \bar{\ell}^q} \;\leq\; \sqrt[q]{n_{2BP}^q + \ell_{2BP}^q} \;\leq\; (1 + \mu)\sqrt[q]{(N^* + k)^q + \bar{\ell}^q}$$

by expanding the absolute value and rearranging the norm terms within R_1. The left inequality holds for any $\mu \geq 1$. As previously recalled, $\ell \leq \tau n$ for any 2BP-DD solution: hence $\ell_{2BP}^q \leq \tau n_{2BP}^q$. Bounding n_{2BP} via the 1-approximation ratio of $\mathcal{H}_{2BP}$ implies, as observed, $n_{2BP} \leq 2(N^* + k)$: the right inequality then reduces to

$$\sqrt[q]{(1 + \tau^q) n_{2BP}^q} \;\leq\; \sqrt[q]{2^q (1 + \tau^q)(N^* + k)^q} \;\leq\; (1 + \mu)\sqrt[q]{(N^* + k)^q + \bar{\ell}^q}$$

Without loss of generality, we set $\bar{\ell} = 0$ to tighten the right-hand side. Indeed, any valid approximation result still holds for any $\bar{\ell} \geq 0$. Then, the inequality becomes

$$\sqrt[q]{2^q (1 + \tau^q)(N^* + k)^q} \;\leq\; (1 + \mu)\sqrt[q]{(N^* + k)^q}$$

and the result is easily derived. □

Proposition 6. *$\mathcal{H}_{2BP}$ is R_2-approximated for 2BP-DD under $\|.\|_q$ with*

$$\mu = \sqrt[q]{1 + 2^q \tau^q}$$

for any $\tau \geq 1$.

Proof. Adapting (2) with $i = 2$ to our case we obtain

$$\sqrt[q]{|n_{2BP} - N^* - k|^q + |\ell_{2BP} - \bar{\ell}|^q} \;\leq\; \mu \sqrt[q]{(N^* + k)^q + \bar{\ell}^q} \tag{3}$$

Using the 1-approximation ratio of $\mathcal{H}_{2BP}$, we have

$$-(N^* + k) \;\leq\; n_{2BP} - (N^* + k) \;\leq\; (N^* + k)$$

which implies

$$|n_{2BP} - (N^* + k)|^q \;\leq\; (N^* + k)^q$$

Plugging this inequality into (3) we get

$$\sqrt[q]{(N^* + k)^q + |\ell_{2BP} - \bar{\ell}|^q} \;\leq\; \mu \sqrt[q]{(N^* + k)^q + \bar{\ell}^q}$$

Now two possibilities arise:

1. if $\ell_{2BP} < \bar{\ell}$, then clearly $(\bar{\ell} - \ell_{2BP})^q \leq \bar{\ell}^q$ and the inequality holds for any $\mu \geq 1$.
2. if $\ell_{2BP} \geq \bar{\ell}$, then we use $\bar{\ell} = 0$ to tighten the inequality; then raising a power of both sides we obtain:

   $$\ell_{2BP}^q \;\leq\; (\mu^q - 1)(N^* + k)^q$$

   Bounding ℓ_{2BP}^q as in Proposition 5, we further get

   $$|\tau n_{2BP}|^q \;\leq\; 2\tau (N^* + k)^q \;\leq\; (\mu_2^q - 1)(N^* + k)^q$$

   from which the result is straightforward.

□

4. A sequential value correction heuristic

In an optimization model with exponentially many variables, solving a pricing problem is fundamental for generating potentially advantageous columns. In the 0-1 LP formulation we refer to for the 2BP problem, columns correspond to optional bin fillings, and the advantage of choosing a particular filling is measured by summing the *shadow prices* of the items that fill the bin. Such prices are the components of an optimal dual solution that correspond to the items at any iteration of the column generation procedure.

In 2BP, solving the pricing problem amounts to solve a two-dimensional knapsack problem, by far more complex than the one-dimensional knapsack used for pricing in 1BP. To save CPU time, one can avoid the LP that is necessary to resolve in order to get the exact shadow prices, and resort to a heuristic evaluation of their values, called *pseudo-pricing*, that we will soon describe. The idea behind a sequential value correction (SVC) algorithm for BP is then quite simple: given a *pseudo-price* p_i for each item $i \in \mathcal{I}$, bins are packed sequentially, each one with the items — not yet allocated — that maximize the sum of pseudo-prices. After a solution has been computed, pseudo-prices are conveniently updated and the process iterated until some halting criterion is fulfilled. In our implementation (here named SVC-DD and described by the pseudo-code Algorithm 1), the computation halts when either a convenient number P of solutions has been computed, or both the number of bins and the maximum lateness of the current solution $\mathcal{S}_k$ close the gaps with the relevant lower bounds n_{LB} and ℓ_{LB} (function check_optimality()).

Originally, SVC algorithms were successfully employed in one-dimensional bin-packing and cutting stock problems [4], then in strip packing [5] and recently in 2BP [10]. Our implementation differs from those proposed in [5, 10], mainly in how bins are filled and pseudo-prices updated. Moreover, in our case the output is a set of reciprocally non-dominated solutions in terms of the number N of bins filled and the maximum lateness $L_{\max}$.

Besides the efficiency and quality of solutions computed by SVC-DD (see Section 6) it is worth noting that SVC-DD does not use any parameter except the two (P and P_{inner}) for controlling the total number of solutions generated. Therefore, differently from the algorithms for 2RBP-DD proposed in [6] and [21], it needs neither a demanding preliminary parameter tuning nor a sensitive analysis.

Let $\mathcal{D}$ be the generalized item set that for 2OBP-DD coincides with $\mathcal{I}$, while for 2RBP-DD also includes the 90° rotated items $h_i \times w_i$ for every $i \in \mathcal{I}$. Moreover, let n_{LB} and ℓ_{LB} be lower bounds to N and $L_{\max}$, respectively.

Algorithm 1 SVC-DD

Input: $\mathcal{D}, W, H, n_{LB}, \ell_{LB}$
Output: solutions $\mathcal{S}_1, \ldots, \mathcal{S}_P$
  $\mathbf{p} \leftarrow$ init_prices$(\mathcal{D}, W, H)$
  for $k \leftarrow 1, \ldots, P$ **do** // compute P solutions
    $\mathcal{S}_k \leftarrow \emptyset$
    $\bar{\mathcal{D}} \leftarrow \mathcal{D}$
    repeat // compute solution $\mathcal{S}_k$
      $\mathcal{B} \leftarrow \emptyset$
      $[S, l(r_{min}), q(r_{min})] \leftarrow$ init_skyline()
      repeat // fill bin $\mathcal{B}$
        $\tilde{\mathcal{D}} \leftarrow$ select_items$(\bar{\mathcal{D}}, l(r_{min}), q(r_{min}))$
        $\mathcal{K}^* \leftarrow$ KP_Conflict$(\tilde{\mathcal{D}}, l(r_{min}), q(r_{min}))$
        $\mathcal{B} \leftarrow$ add_items$(\mathcal{K}^*)$
        $[S, l(r_{min}), q(r_{min})] \leftarrow$ update_skyline$(\mathcal{K}^*)$
        $\bar{\mathcal{D}} \leftarrow$ remove_items$(\mathcal{K}^*)$
      until $(l(r_{min}), q(r_{min})) \neq (W, H)$
      $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{\mathcal{B}\}$
    until $(\bar{\mathcal{D}} \neq \emptyset)$
    if check_optimality$(\mathcal{S}_k, n_{LB}, \ell_{LB})$ **then**
      return
    end if
    if $(i-1) \equiv 0(\mod P_{inner})$ **then**
      $\delta \leftarrow$ random$(0, 1]$
    end if
    $\mathbf{p} \leftarrow$ update_prices$(\mathcal{S}_k, \mathbf{p}, n_{LB}, \ell_{LB}, \delta)$
  end for

4.1. Pseudo-price update

Due to the bi-criteria objective, pseudo-prices reflect both the packing and scheduling quality of the solutions generated. Assume items sorted by non decreasing due-dates, i.e., $d_1 \leq \ldots \leq d_m$. Initially, the function init_prices() sets the pseudo-prices to

$$p_i := \frac{WH}{d_i + 1} + w_i h_i \qquad i \in \mathcal{D} \tag{4}$$

where the first term promotes urgent items, while the second increases the price of large items.

After the current solution $\mathcal{S}_k$ has been computed, pseudo-prices are updated in function update_prices(). The formulas we use to modify pseudo-prices are designed to jointly exploit information related to i) the instance (e.g., number of items, bin area, lower bounds), ii) features of the current solution (e.g., residual space in bins, completion time of the tardiest item). At each step, the information is merged so that pseudo-price modifications are adequately limited. First, the value of p_i, $i \in \mathcal{D}$, is increased by a factor accounting for the unused space of the current solution $\mathcal{S}_k$:

$$p_i := p_i \left(1 + \delta \cdot \frac{n_{LB}}{m}\right)^{\frac{\omega(i)-\bar{\omega}}{mWH}} \tag{5}$$

where $\omega(i)$ is the residual space in the bin where item i is packed, and $\bar{\omega} = \frac{WHn_{LB}-\sum_{i\in D} w_i h_i}{n_{LB}}$ is a lower bound to the overall relative residual. Since the residual of a bin generally increases with the bin relative position in the sequence, the formula promotes late packed items with bad patterns, thus ensuring a suitable mix of the search space. The quality of each pattern is esteemed by measuring the deviation of $\omega(i)$ from the reference point $\bar{\omega}$, and suitably normalized. The base perturbation is then more relevant for instances whose solutions have a large number of bins, where it is profitable to explore a wide collection of patterns. On the other hand, smaller updates are done in case of many items, where the evolution of pseudo-prices is kept under control to progressively explore the search space. The exponent and the constant term in the base are small fractional numbers that prevent pseudo-prices from diverging quickly.

Also, the pseudo-price of the tardiest item h (and possibly of its rotated counterpart) is further increased by a factor proportional to the current maximum lateness:

$$p_h := p_h \left\{1 + \left(1 + \delta \cdot \frac{\ell_{LB}}{\tau \cdot n_{LB}}\right) \cdot \left[\frac{(C_h - \ell_{LB})/\tau}{d_h + 1}\right]\right\}. \tag{6}$$

where as usual C_h is the completion time of h in the current solution $\mathcal{S}_k$.

The fractional factor $\left(1 + \delta \cdot \frac{\ell_{LB}}{\tau \cdot n_{LB}}\right) > 1$ is generally very close to 1, whereas the term $\left[\frac{(C_h-\ell_{LB})/\tau}{d_h+1}\right]$ linearly grows with the current lateness of item h. The pseudo-price correction is larger when h has a close due date

and is packed in a late bin, and is in general emphasized for instances with high delay values on a limited number of expected bins. Again, numbers are so designed as to induce relatively small increments of the current pseudo-price.

Coefficient δ is randomly selected in $(0, 1]$ to perturb pseudo-price dynamics, and is renewed each P_{inner} main iterations.

Update (5) is omitted if the number n of bins in the current solution reaches the corresponding lower bound, thus being certified optimal.

4.2. Bin filling

The bin filling procedure here described inserts items of $\mathcal{D}$ into the current bin $\mathcal{B}$ in a bottom-up fashion by iterated solution of one-dimensional knapsack problems. At the end of each step, the upper borders of the items inserted form a collection of horizontal segments, the *roofs*, that altogether form a bin *skyline*.

Focusing on a generic step of the procedure, let $\bar{\mathcal{D}} \subseteq \mathcal{D}$ be the set of items not yet allocated, and let $S = \langle r_1, \ldots, r_{|S|} \rangle$ (a bin *skyline*) be an ordered list of roofs associated with the bin $\mathcal{B}$ being filled, see Figure 2-3. Each roof r_j of the bin skyline S is described by a length $l(r_j)$ and a height $q(r_j)$ measured from the bottom of the bin; endpoints of adjacent roofs sharing the projection on the bin bottom edge are connected by a vertical segment that we call a *wall*; the leftmost (the rightmost) roof has the left (the right) wall coincident with the bin vertical edges. The bin skyline defines a border within $\mathcal{B}$ above which unpacked items can be placed (see Figure 2-3; we assume w.l.o.g. that adjacent roofs have distinct heights: consecutive roofs at the same height are merged into a single one).

At the beginning, i.e., when $\mathcal{B}$ is empty, the function init_skyline() sets the bin skyline to a single "ground" roof r_1 with $q(r_1) = 0$ and $l(r_1) = W$; the bin filling process iteratively places unpacked items on the lowest roof of the bin skyline. Specifically, let r_{min} be the roof at minimum height in the current bin skyline

1. If no item can be placed on top of r_{min} — viz., $h_i > H - q(r_{min})$ for each $i \in \bar{\mathcal{D}}$ — the procedure ends.
2. Otherwise (Figure 2) function KP_Conflict() defines and solves a 0-1 one-dimensional knapsack problem (KP), with knapsack capacity $l(r_{min})$ and items in $\tilde{\mathcal{D}} = \{i \in \bar{\mathcal{D}} | w_i \leq l(r_{min}) \text{ and } h_i \leq H - q(r_{min})\}$:

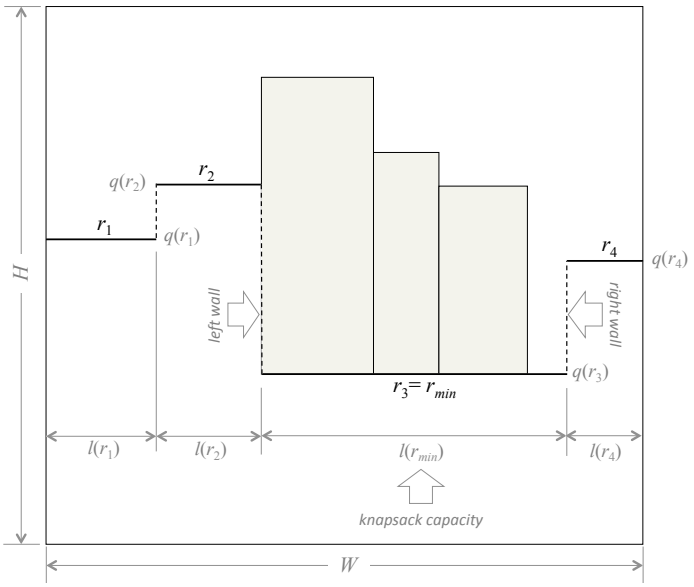

Figure 2: A bin skyline S consisting of 4 roofs $r_1, \ldots, r_4$; the length of the roof at minimum height, that is $r_{min} = r_3$, defines the capacity of the relevant knapsack; the items of a knapsack solution (grey rectangles) are ranked by non increasing heights, starting from the highest wall of r_{min} — in this case, the left one.

the items of an optimal solution $\mathcal{K}^* \subseteq \tilde{\mathcal{D}}$ of KP are placed on top of r_{min}, sorted by non increasing heights and placed in this order starting from the highest among the delimiting walls.

3. Then (Figure 3) function update_skyline() updates the bin skyline by adding a new roof for each item of $\mathcal{K}^*$ and possibly enlarging one roof adjacent to r_{min} of an amount corresponding to the unused knapsack capacity. The bin skyline update involves a new r_{min}, and the procedure is iterated going back to step (1).

Items in $\mathcal{K}^*$ (and possibly their rotated counterparts) are removed from $\bar{\mathcal{D}}$ and the process iterated.

For 2OBP-DD, KP is a standard 0-1 knapsack problem. For 2RBP-DD, however, it becomes a knapsack problem with compatibility constraints: in fact, at most one item of $\mathcal{D}$ out of $w_i \times h_i$, $h_i \times w_i$ can be allocated to the knapsack. In our implementation, we use a standard knapsack solver and handle conflicting items heuristically: while the solution $\mathcal{K}^*$ contains

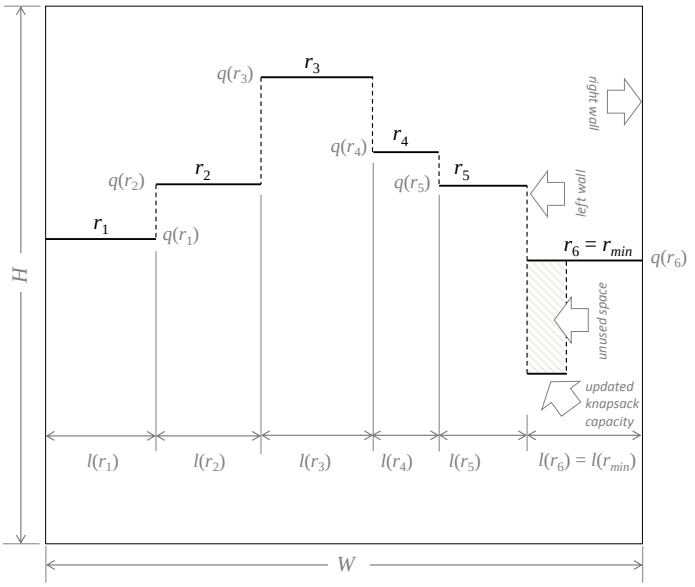

Figure 3: Bin skyline of Figure 2 updated by placing the three items of the knapsack solution: the original roof r_3 is replaced by a new r_3, plus r_4 and r_5, with lengths and heights defined according to the items added; r_6 is then enlarged to its left for a length corresponding to the unused space (dashed area). After update, it turns out $r_{min} = r_6$.

conflicting items, we remove from $\tilde{\mathcal{D}}$ the one with smallest pseudo-price per surface unit, i.e., a conflicting item i with minimum $p_i/(w_i h_i)$; then we solve KP again. Although potentially demanding in terms of CPU time, such a simple strategy performs very well since conflicts seldom occur (roughly, in 4% of the problems solved).

5. Dual bounds

The strong correlation determined by Assumption 2 between the minimum number N of bins and the minimum $L_{\max}$ still holds for dual bounds, and lower bounds to N can be exploited to build lower bounds to $L_{\max}$ (see [6]).

A simple lower bound for 2OBP can easily be derived by considering the two one-dimensional bin packing problems with bins of sizes W and H and items of sizes $w_1, \ldots, w_m$ and $h_1, \ldots, h_m$, respectively. Tighter bounds can be derived by using *dual feasible functions* (DFF) as described in [1]:

Definition 1. *A discrete function $f : [0, C] \to [0, C']$ (with C, $C' \in \mathbb{Z}$) is*

dual feasible *if*

$$\sum_{x \in S} x \leq C \Rightarrow \sum_{x \in S} f(x) \leq f(C) = C'$$

for any finite discrete set S*.*

Although such bounds are not valid for 2RBP, which is a problem less constrained than 2OBP, they can still be exploited provided that the instances of 2RBP are adequately relaxed, as proposed in [8]: given an instance of 2RBP with item set $\mathcal{I}$ and square bins, construct an instance of 2OBP with the item set $\mathcal{D}$ obtained as described in §4, i.e., by expanding $\mathcal{I}$ with the 90° rotated counterpart of each item. Then, for any lower bound N^O_{LB} to the 2OBP-optimum of $\mathcal{D}$, $N^R_{LB} = \left\lceil \frac{N^O_{LB}}{2} \right\rceil$ is a lower bound to the 2RBP-optimum of $\mathcal{I}$. The proposed approach can be employed also when bins of $\mathcal{I}$ are rectangular, by transforming them into squares and introducing an appropriate number of dummy items to avoid over-relaxation. However, the authors of [8] show that N^R_{LB} dominates known lower bounds of 2RBP only in the case of square bins.

A lower bound to $L_{\max}$ proposed in [6] is based on the joint consideration that (*i*) the EDD rank $\sigma = \langle 1, \ldots, m \rangle$ is optimal for $\min L_{\max}$ 1-machine scheduling, (*ii*) the minimum time at which the k-th item in σ is scheduled in a combined packing-scheduling problem is at least τ times the minimum number of bins necessary to accommodate the first k items in the rank, $\mathcal{F}_k = \{1, \ldots, k\}$, or anyway no less than a lower bound N^k_{LB} to this value. Therefore, a lower bound to L_k is

$$L^k_{LB} = \tau N^k_{LB} - d_k$$

and the bound to $L_{\max}$ is computed as

$$L_{LB} = \max_{k=1,\ldots,m} \{L^k_{LB}\} \tag{7}$$

The tighter the N^k_{LB}, the more L_{LB} becomes effective, and we note here that L^k_{LB} can be further tightened under simple conditions, independently on the quality of the N^k_{LB} employed, i.e., even when N^k_{LB} coincides to the optimal number of bins. In fact, L_{LB} is obtained by computing all the terms appearing in (7), as if item k were packed in the N^k_{LB}-th bin, even when no feasible solution with $L_{\max} = L^k_{LB}$ can place k in that bin.

Table 1: Data for an example of lower bound to $L_{\max}$.

item	length l_i	height h_i	due date d_i
1	1	1	1
2	1	2	1
3	1	3	1
4	1	4	1
5	1	6	4
6	1	7	4
7	1	8	4
8	1	9	4

Consider the following simple example: there are $m = 8$ items, with sizes and due dates as in Table 1, to be packed into bins of size $L \times H = 1 \times 10$ and scheduled in time slots of length $\tau = 1$. Items can be packed using four bins, thus $N^m_{LB} = 4$ and formula (7) returns $L_{LB} = 0$. No EDD rank can be packed in four bins, though (see Figure 4): either the completion time increases by one time unit as in the leftmost schedule; or, as in the rightmost, compressing the items in less bins has the effect of 'dragging' urgent items towards the end of the schedule, so making an EDD rank impossible. The shortest EDD schedule and the tightest packing (which does not admit an EDD schedule) respectively give $L_{\max} = 1$ and $L_{\max} = 3$, hence $L_{\max}$ cannot be less than 1 time unit.

Generalizing the above example, we can improve L_{LB}. Let $\sigma = \langle 1, \ldots, m \rangle$ be an EDD rank. Call items h and k *compatible* if they can share the same bin and, for $h < k$, let $\bar{N}^i_{LB}$ be a lower bound to the tightest packing of

- $\{1, \ldots, i\}$ for $1 \leq i \leq h$ (that is $\bar{N}^i_{LB} = N^i_{LB}$)
- $\{1, \ldots, i, k\}$ for $h < i < k$

Finally, let $N^{i,\text{EDD}}_{LB}$ be a lower bound to the tightest packing computed on the first i items of the EDD rank described by σ, i.e., the item completion times defined by the packing are compliant with the EDD rank expressed by σ. Then

Proposition 7. *Suppose that, for some $k \leq m$, $N^{k,\text{EDD}}_{LB} > N^k_{LB} = N^{k-1}_{LB}$. Let h be the closest item that is compatible with k in the EDD rank σ, and*

$$L' = \max \left\{ \begin{array}{ll} \tau N^i_{LB} - d_i & 1 \leq i \leq h \\ \tau \bar{N}^i_{LB} - d_i & h < i < k \end{array} \right\}$$

$$L'' = \max \left\{ \begin{array}{ll} \tau N^i_{LB} - d_i & 1 \leq i < k \\ \tau (N^i_{LB} + 1) - d_i & i = k \end{array} \right\}$$

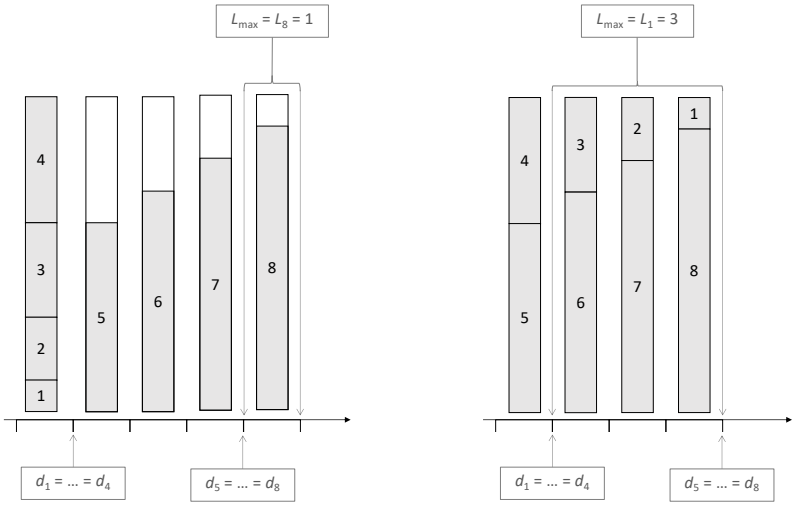

Figure 4: $L_{\max}$ obtained with the tightest packing of an EDD rank (left) and the tightest packing at all.

Then

$$\bar{L}^k_{LB} = \min\{L', L''\} \tag{8}$$

is a lower bound to $L_{\max}$.

Proof. If $N^k_{LB} = N^{k-1}_{LB}$, then the tightest packing of $\{1, \ldots, k\}$ necessarily places the k-th element with one compatible with k, say $h < k$. If no such item exists, then $N^{k,EDD}_{LB} > N^k_{LB}$ implies that N^k_{LB} can be lifted by one bin:

$$L_{\max} \quad \geq \quad \tau(N^i_{LB} + 1) - d_i \qquad\qquad i = k$$

Otherwise, consider the sequence $\bar{\sigma} = \langle 1, \ldots, h-1, h, k, h+1, \ldots, k-1 \rangle$ which is compliant with this packing. As h and k are paired in the same bin, they can regarded as a single item due by $\min\{d_h, d_k\} = d_h$, therefore $\bar{\sigma}$ is an EDD rank on $k-1$ items. Then, using (7) and recalling the definition

of $\bar{N}^i_{LB}$ we obtain:

$$\begin{aligned} L_{\max} &\geq \tau N^i_{LB} - d_i && 1 \leq i \leq h \\ L_{\max} &\geq \tau \bar{N}^i_{LB} - d_i && h < i \leq k-1 \end{aligned}$$

where, to alter item completion times as least as possible, h is the closest item in σ that is compatible with k. Note that h and k are completed at the same time and $d_h \leq d_k$: so we do not need to specify a bound for $i = k$.

In alternative, we can bound L_i using σ, with N^i_{LB} as an (optimistic) completion time of $\{1, \ldots, i\}$ for $i < k$, and $N^{k,\text{EDD}}_{LB} \geq N^k_{LB} + 1$ for $i = k$:

$$\begin{aligned} L_{\max} &\geq \tau N^i_{LB} - d_i && 1 \leq i < k \\ L_{\max} &\geq \tau (N^i_{LB} + 1) - d_i && i = k \end{aligned}$$

Choosing the case for which $L_{\max}$ is reduced most, we finally obtain (8). □

By Proposition 7 we get an improved lower bound $\bar{L}_{LB}$ which is generally more effective in presence of large items. At the expense of additional CPU time, the result can be further reinforced by considering the items with both sides exceeding half the sides of the bin:

$$\mathcal{I}_{\mathcal{L}} = \{i \in \mathcal{I} : \min(w_i, h_i) > \max(\frac{W}{2}, \frac{H}{2})\}$$

For the 2BP, pairs in $\mathcal{I}_{\mathcal{L}}$ cannot be packed in the same bin. When computing L^i_{LB}, for $1 \leq i \leq h$ the bound on the number of bins N^i_{LB} can be tightened by assuming items in $\mathcal{I}_{\mathcal{L}}$ as packed in different bins, thus reducing the area available to pack the remaining items in such bins. This can be straightforwardly done by imposing that N^i_{LB} cannot be smaller than the number of items $j \in \mathcal{I}_{\mathcal{L}}$ such that $1 \leq j \leq i$. Alternative (and possibly more effective) techniques can be designed by exploiting dual bounds for the bin packing problem with variable stock sizes, thus computing N^i_{LB} in a scenario in which bins have different sizes [9].

As described in [21], a lower bound to $L_{\max}$ can also be obtained by relaxing the geometrical constraints of a MILP formulation for the maximum lateness minimization into DFFs-based inequalities which ensure that the total area of items inside each bin is dual feasible. Lower bounds obtained in this way are generally good, see Section 6, but the approach calls for the

solution of a MILP which is usually time consuming and poorly scalable.

We close this section by observing that the specific structure of L_{LB} also allows to infer the cardinality of the Pareto frontier. Let $\bar{x} = (\bar{n}, \bar{\ell})$ be a feasible solution.

Proposition 8. *If $\bar{n} = N_{LB} + h$ and $\bar{\ell} \leq \max\{L_{LB}, \tau(\bar{n} + k) - d_n\}$, with $h \geq 0$ and $k \geq 1$, then the set of strictly non-dominated solutions contains at most $h + k$ points.*

Proof. Let $\tilde{x} = (\bar{n} + k, \tilde{\ell})$ be a feasible solution. $\tilde{\ell} \geq \tau(N_{LB} + h + k) - d_n$ by (7) therefore $\tilde{x}$ is dominated by $\bar{x}$, and there are at most $h+k$ non-dominated solution with less than $\bar{n} + k$ bins. □

6. Computational results

The SVC-DD algorithm was coded in C++ and compiled with Microsoft® C/C++ Optimizing Compiler (version 19.11.25447). Parameters P and P_{inner} were set to 10^3 and 10^2, respectively. The n_{LB} and ℓ_{LB} that appear in formulas (5), (6) and in function check_optimality(), are respectively set to N^R_{LB}, and to $\bar{L}_{LB}$ (computed employing N^R_{LB}).

Tests were performed on a Intel® Core(TM) i7-7500U 2.90 GHz with 16Gb RAM, and were conducted on two sets $\mathcal{I}_R$ and $\mathcal{I}_B$ of instances. All the tables detailing the instance features and the numerical results are reported in the Appendix.

The former set derives from 28 industrial orders collected by SCM [24], an Italian group with a major international reputation in the sector of cutting machines. The problems considered have up to 360 items with sizes $w_i \in [30, 2928]$, $h_i \in [24, 1820]$ and rectangular large bins with sizes $W \in [2200, 5600]$, $H \in [1163, 2100]$. For each instance in this set, Table 2 in the Appendix reports the number m of items, the sizes W and H of the bin, the ratio $\frac{w_i h_i}{WH}$ between item and bin areas (*size factor*: min, max, and mean value) and the ratio $\frac{w_i}{h_i}$ between item widths and heights (*shape factor*: min, max, and mean value). Instances show quite heterogeneous size and shape factor ratios and, neglecting the distinction induced by m, can be grouped by similarity considering the distances $\max\{\frac{w_i}{h_i}\} - \min\{\frac{w_i}{h_i}\}$ and $\max\{\frac{w_i h_i}{WH}\} - \min\{\frac{w_i h_i}{WH}\}$, with the ratio mean value as reference point. This range seems reasonable as the minimum ratios are close enough, with only few exceptions (r_{11}, r_{16} and r_{24}). Instances r_3, r_5, r_6, r_8, r_{13} and r_{14}

share a small variability of both size and shape factors, within respective interval lengths 0.2 and 10. All those instances have mean $\frac{w_i}{h_i} \geq 2.2$, except r_5 and r_{13}, where it appears lower (1.51 and 1.41). The subset r_2, r_4, r_9, r_{10}, r_{11}, r_{16} and r_{24} presents shape factors spread in the same small min-max interval, whereas the min-max distance of size factors moves on average in a range between 0.2 and 0.5. Instance r_{10} consists of relatively tiny items, as testified by a mean size factor of 0.08, while for r_{11}, r_{16} and r_{24} the shape factor $\frac{w_i}{h_i}$ has a quite large average, above 3.4. Affinities can be observed in the subgroup r_7, r_{12}, r_{15}, r_{18}, r_{19} and r_{20} where, given a size factor variability analogous to the previous subset (e.g., r_2 and r_4), the min and max shape factors define ranges of moderate width (between 10 and 30). Here the mean size factors of r_7 and r_{12} are quite large (0.24 and 0.13), and the mean shape factors of r_{15} and r_{20} are relatively small (2.33 and 2.56). Finally, some specific affinity based on min-max ranges can be grasped in the sets $\{r_{22}, r_{23}\}$, $\{r_1, r_{25}\}$, $\{r_{21}, r_{26}, r_{27}\}$. The first set presents a moderate size factor variability, while $\frac{w_i}{h_i}$ spreads across a large min-max interval (> 30); however, r_{22} and r_{23} disagree on the average shape factor (3.85 of r_{22} vs. 6.04 of r_{23}). The second set is characterized by a size factor in a wide min-max interval (above 0.5), a shape factor within small boundaries (below 10) and a good affinity on the average measures. The instances in the last set share large min-max intervals on both factors, with r_{26} showing a lower mean $\frac{w_i}{h_i}$ of 4.27 than r_{21} and r_{27} having a factor larger than 5.30.

The latter set $\mathcal{I}_B$ of instances was kindly provided by the Authors of [6], who added due dates to five hundred benchmark instances reported in [7] (classes I-VI) and [17] (classes VII-X). Each class I to X includes fifty instances grouped by ten into five subsets: each subset has $m \in \{20, 40, 60, 80, 100\}$ items and square bins. For each class, Table 3 in the Appendix reports the bin size, the assortment of item sizes and, according to [6], the number of instances with $N_{LB} > 1$ (column "# inst."). The other entries in row i give an indication of the frequency of item sizes of class i: for example, 0.7 in the last row and column means that 70% of the items were generated with w, h uniformly chosen in $[1, \frac{1}{2}W]$ and $[1, \frac{1}{2}H]$, respectively.

Due-dates for both $\mathcal{I}_R$ and $\mathcal{I}_B$ instances were generated by randomly pick integers in the interval $[\tau + 1, \tau\beta N_{LB}]$ with $\beta \in \{0.6, 0.8, 1.0\}$ and the N_{LB} employed in [6]: hence $\mathcal{I}_R$ and $\mathcal{I}_B$ consist of three different due-dates groups A, B, C, amounting on the whole to eighty-four $\mathcal{I}_R$-instances and

one thousand five hundred $\mathcal{I}_B$-instances.

SVC-DD was specifically designed for 2RBP-DD but, skipping (6) and omitting the due-date related term in (4), it can also be configured as a quite well-performing heuristic for 2BP, see also [19]. Interestingly, this algorithm turns out to be competitive against heuristics that were specifically designed for bin packing. Table 4 in the Appendix shows the behaviour of SVC-DD when used to solve a traditional bin packing problem: the table reports, for each instance class, number of optima found (col. "# opt") and average running times (col. "CPU"). Also, the table compares the mean number $\bar{n}$ of bins in the best solutions found by SVC-DD to that employed by the heuristic SVC2BPRF proposed in [10], and to the best $\bar{n}$ achieved in a range of benchmark algorithms chosen from the literature in [10] for comparison (see column "best H"). Solutions are obtained in similar running times (0.90 seconds on average). In two cases (classes II and IV) the improvement on SVC2BPRF is strictly positive, in one case (class IV) SVC-DD even outperforms all the benchmark heuristics.

6.1. Comparison to other approaches

In the first part of our experiments, we compare SVC-DD on data-set $\mathcal{I}_B$ to:

- the multicrossover genetic heuristic (MXGA) proposed in [6];
- the hybrid constraint and integer linear programming approach (CPMIP) described in [21].

Both algorithms above are specifically designed for 2RBP-DD. For a fair comparison of results, we use the same value of τ (= 100) and solution analysis as in [6, 21], adopting the performance indicators (primal-dual gaps)

$$G_N = 100\frac{(n - N_{LB})}{N_{LB}} \qquad\qquad G_L = 100\frac{(\ell - L_{LB})}{L_{LB}}.$$

where (n, ℓ) is a solution with n bins and maximum lateness ℓ, and N_{LB} and L_{LB} are the dual bounds used in [6].

In particular, for each instance we take a solution (π_n) achieving the minimum N and one (π_ℓ, possibly different from π_n) with minimum $L_{\max}$, and then measure their quality by G_N and G_L.

SVC-DD vs. MXGA

Table 5 in the Appendix shows the aggregated results of MXGA and SVC-DD. Each row (i.e., for each class of instances and due-date groups A, B, C) reports the gaps G_N and G_L of π_n and π_ℓ averaged on the 50 instances of each class.

Numerical figures show that SVC-DD largely improves the mean gaps in all instances, with just one exception (G_N of class VII, group C): accordingly, the overall gap percentage improvement $(G_\bullet^{\text{MXGA}} - G_\bullet^{\text{SVC-DD}})/G_\bullet^{\text{MXGA}}$, averaged on all the groups of instances, is $G_N = 26.7\%, G_L = 38.1\%$ for solutions π_n, and $G_N = 26.1\%, G_L = 41.3\%$ for solutions π_ℓ. In particular, on classes II, IV and VI the overall gap percentage improvement achieves the widest values, with SVC-DD performing at least 75.8% better than MXGA. According to Table 3 (see the Appendix), SVC-DD generally obtained its best results in instances with relatively small items. Indeed, SVC-DD reaches very limited gaps in classes II, IV, VI and IX, with $G_N \leq 2\%$ for all groups and $G_L \leq 3\%$ for group A. The gap G_L grows up to 6.2% in group B and further arrives up to 15.2% in group C, with the exception of class VI where $G_L = 58.4\%$. In addition, SVC-DD finds solutions with a minimum number of bins in all the instances of classes II.

On the other hand, MXGA performs better in some cases: solution π_n in class I for groups A and B, values of G_N in classes I, VII and VIII for example. Still, the overall gap is less wide, and MXGA is at most 23.5% better than SVC-DD. Actually, classes VII and VIII correspond to the highest G_N values reached by SVC-DD, which are always above 7.5%, up to 12.3%. In the same classes, also the G_L values are meaningful, growing from 20.6% for π_l in group A to 242.4% for π_n in group C. Higher gaps are reached in class X only, topping 275.6% for π_n. Nevertheless, these large G_L gaps are not particularly representative (see Section 6.3) and remain significantly smaller than those obtained by MXGA.

About CPU time, though requiring the repeated solution of 0-1 knapsack problems, SVC-DD has a mean running time of 1.46 sec. This value is two order of magnitude less than the CPU times reported in [6], a speedup that cannot be just ascribed to hardware configuration.

SVC-DD vs. CPMIP

SVC-DD can be just partially compared to CPMIP, since [21] focuses on

$L_{\max}$ minimization and therefore gives gaps G_L but no detail on packing quality. Thus, we focus on the comparison of G_L for solutions π_ℓ.

Table 6 in the Appendix reports the percentage gaps G_L of the primal solutions π_ℓ computed by MXGA (same as the fifth column of Table 5), CPMIP and SVC-DD (same as the ninth column of Table 5). The figures show that SVC-DD provides mean percentage gaps always better than CPMIP except for class IX, where the gaps of groups A and B coincide. In the other classes, the percentage gap improvement $(G_L^{\text{CPMIP}} - G_L^{\text{SVC-DD}})/G_L^{\text{CPMIP}}$ spreads from 15.8% up to 82.6% in group A, reducing in group B from 7.8% to 56.6% and in group C from 8.2% to 57.3%. The widest gap improvements are generally observed in classes II, IV and VI (respectively 57.5%, 61.8% and 49%, averaged across all the groups), while the smallest are found in classes I, V (11.1% and 14.3% averaged across all the groups) and III in group C only (8.2%). Summarizing, the percentage gap improvement averaged on the classes is of 34.4% on group A, 24.6% on group B and 22.4% on group C.

As a final remark, SVC-DD was able to certify the optimality of L_{max} in 620 cases out of 1500 vs. 586 cases in which CPMIP does the same — but with a mean running time of 30.2 seconds, roughly twenty times that of SVC-DD. The certification phase makes use of the best bound configuration for each algorithm, i.e. N_{LB}^R and $\bar{L}_{LB}$ for SVC-DD and the tightest lower and upper bounds for CPMIP discussed in [21].

6.2. Improved dual bounds

The Authors of [21] propose two different lower bounds to $L_{\max}$, the tighter of which computed via a Mixed Integer Linear Program (MILP). In their experiments, for 256 instances out of 1500 the MILP was not able to return a valid bound within one hour of CPU time. A valid bound for the remaining 1244 instances was computed within a mean running time of 74.91 seconds, and in 361 cases it strictly improved (by 31.30% on average) that reported in [6]. Nonetheless, the bounds proposed in [21] do not dominate L_{LB} as computed in [6] and therefore do not dominate $\bar{L}_{LB}$.

In our tests, the improvements (both individual and combined) achieved via Proposition 7 and by employing the bin packing dual bound N_{LB}^R are the following. $\bar{L}_{LB}$ (computed without employing N_{LB}^R) resulted tighter than L_{LB} (7) in 39 instances (2.60% of the cases), with an improvement on

this subset that ranges from 0.36% to 223.53%, 22.25% on average. Specifically, the bound was improved in eleven instances of group A (2.20% of the cases), with an average (minimum, maximum) improvement of 4.36% (0.36%, 11.41%). In both groups B and C the favorable cases rise to fourteen (2.80% of the cases), and the improvement magnitude increases as well: in group B, it ranges between 1.31% and 63.28%, with a mean improvement of 16.12%; in group C, the minimum improvement was 5.43%, the maximum 223.53%, and 42.44% on average. Looking at instance classes, eight cases of improvement were observed across all groups, respectively in class I and III (mean improvement of 9.92% and 31.02%), fifteen cases in class V (improvement of 13.14% on the average), two in class VII (7.35% of mean enhancement) and six in class X (54.75% improvement on the average). Due to the rationale of Proposition 7, which implicitly relies on item incompatibility (see §5), improvements are achieved on instance classes with relatively large items; in fact, in classes II, IV and VI all items are compatible and the proposition ineffective (see Table 3 in the Appendix).

On the other hand, in 505 cases the bound in [6] strictly improves (by 28.07% on average) when computed using N^R_{LB}. Via Proposition 7, this bound is further improved in a small fraction of cases: for 2RBP-DD, 0.67% of the instances showed an improvement from 0.38% to 35.42%, 11.44% on average. The mean improvement respectively reaches 1.05%, 16.08% and 9.82% in group A, B and C. Averaged across instance classes, the improvement gets 10.46% in class I, 10.08% in class III, 7.87% in class V, 6.07% in class VIII and 35.42% in class X. For 2OBP-DD, an improvement is recorded in 1.26% of the instances, ranging from 0.27% to 14.53%, 4.25% on average. In Group A the bound was enforced on average by 1.33%, in group B by 3.62% and in group C by 6.42%. Such improvements were achieved on average across all groups as 7.29% in class I, 3.39% in class III, 3.52% in class V, 1.71% in class VIII and 8.98% in class X. Again, improvements emerge more often for instances with relatively large items.

Details on the gaps obtained by using $\bar{L}_{LB}$ (computed by employing N^R_{LB}) are reported in Table 7. By comparing such gaps with those reported in Table 5, it emerges that the gaps of solutions π_n are lowered by 35.8% (G_N) and 30.3% (G_L) in the overall, and the gaps of π_ℓ by 30.1% (G_N) and 34.3% (G_L). Thanks to the improved bound, SVC-DD is proved to find an *ideal* point (a solution that achieves both absolute minima N and $L_{\max}$, see also

§6.3 below) in all the instances of class IX. In addition, the largest gap reduction, averaged on all groups, is observed in classes I, III and V (from 51.4% to 88.4%, 42.9% to 57.0%, 49.6% to 65.3%), whilst no improvement was achieved on classes II, IV and VI. The overall mean improvement ranges from 30.1% to 36.3% for group A, from 27.5% to 35.3% for group B and from 29.9% to 35.1% for group C. All the lower and upper boundaries are given by the improvements of π_n on G_L and G_N, respectively.
Averaged on due-date types, the best gaps G_L obtained in [21] exploiting the best known lower bounds are 11.9% for due-date group A, 20.5% for group B and 68.5% for group C. For SVC-DD, the best gaps G_L are improved to 7.4% in group A, 14.2% in group B and 51.8% in group C.
As a final remark, $\bar{L}_{LB}$ was obtained in 0.63 seconds on average.

6.3. Pareto-analysis

Measuring solution quality by gaps G_N and G_L has inherent limits, as 2BP-DD is a multi-objective problem. Moreover, for relatively large due-dates and τ, G_L results very data-sensitive: for example, with $\tau = 100, L_{LB} = 5$ and $d_i = 95$, gap G_L jumps from 0% (when item i is assigned to the first bin) to 2000% (when i is allocated one bin later). In order to assess the performance of SVC-DD we then preferred two measures, that we derived from papers on multi-objective evaluation: $\bar{R}_1$ and G_A. The former uses the ratio $R_1(x_{\mathcal{H}}, x^*)$ proposed by [12], see §3.1; the latter is a slight modification of the *Space Covered Measure* (SCM) presented by [28]. In both cases we resort to the notion of *ideal* solution value, intended as the point $x_{id} = (n_{id}, \ell_{id}) \in \mathbb{N}^2$ with $n_{id} = N_{LB}$ and $\ell_{id} = L_{LB}$.

Let $\bar{X} = \{\bar{x}_i = (\bar{n}_i, \bar{\ell}_i), i = 1, \ldots, p\}$ be the set of non-dominated solutions computed through a heuristic for 2BP-DD. Indicator $\bar{R}_1$ is the minimum of $R_1(\bar{x}_i, x_{id})$ computed with norm $\|.\|_2$ among all solution values in $\bar{X}$:

$$\bar{R}_1 = \min_{i \in \bar{X}} R_1(\bar{x}_i, x_{id}) = \frac{|\sqrt{\bar{n}_i^2 + \bar{\ell}_i^2} - \sqrt{N_{LB}^2 + L_{LB}^2}|}{\sqrt{N_{LB}^2 + L_{LB}^2}} \tag{9}$$

While $\bar{R}_1$ is constructed after the differences of solution value norms from the norm of the ideal point, indicator G_A attempts at evaluating the area that underlies the Pareto-frontier (as approximated by solutions in $\bar{X}$). Let $x_{nad} = (N_{UB}, L_{UB})$ be the *nadir* point, where N_{UB} and L_{UB} are valid upper bounds to N and $L_{\max}$ respectively. A frontier is evaluated by two

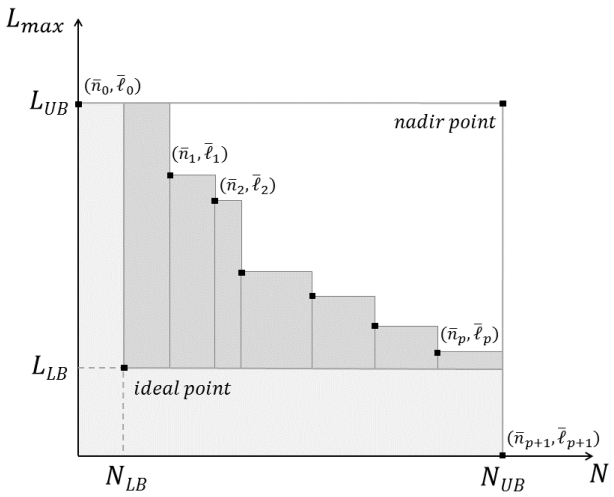

Figure 5: A_{id} = light grey area; $A_{\bar{X}} - A_{id}$ = dark grey area; $\bar{x} = (\bar{n}_i, \bar{\ell}_i), i = 1, \ldots, p$, are the solution values in the heuristic frontier $\bar{X}$.

areas computed as sums of rectangles, see Fig. 5: the first area underlies x_{id} and is given by $A_{id} = N_{LB}(L_{UB} - L_{LB}) + L_{LB}N_{UB}$; the second is associated with $\bar{X}$ and amounts to $A_{\bar{X}} = \sum_{i=0}^{p} \bar{\ell}_i(\bar{n}_{i+1} - \bar{n}_i)$, where solution values are sorted by increasing N and $\bar{n}_0 = \bar{\ell}_{p+1} = 0$, $\bar{\ell}_0 = L_{UB}$, $\bar{n}_{p+1} = N_{UB}$. The quality of a heuristic frontier $\bar{X}$ is then measured through the percent gap:

$$G_A = 100\frac{A_{\bar{X}} - A_{id}}{A_{id}} \tag{10}$$

Gap G_A increases as the heuristic frontier steps further from x_{id}, and takes into account both the quality and cardinality of $\bar{X}$. Note that G_A mainly differs from the SCM of [28] in how areas are computed, but can be employed to evaluate any bi-objective optimization algorithm. Clearly, both $\bar{R}_1$ and G_A equal zero only when x_{id} is the value of a feasible packing, which implies that the optimal frontier consists of just one solution of value x_{id} (proving the existence of such packing is NP-complete).

Since $\bar{R}_1$ is more meaningful when solution values are comparable to each other, the computational results reported next are obtained by setting $\tau = 1$ and normalizing due-dates accordingly.

We conducted the above described Pareto-analysis on both instance sets $\mathcal{I}_B$ and $\mathcal{I}_R$. Table 8 in the Appendix shows the results on $\mathcal{I}_B$ instances. The

table reports $\bar{R}_1$ and G_A for each class and due-date group, with G_A referred to the nadir point $x_{nad} = (n_{EDD}, \tau \cdot n_{EDD})$, see Proposition 2. The table also gives

- how many times ($\#m$) the set $\bar{X}$ of reciprocally non-dominating solutions found by SVC-DD has more than one point,
- how many times ($\#s$) Proposition 8 certifies that the optimal frontier consists of a single solution,
- how many instances ($\#opt$) out of $\#s$ are solved to optimality by SVC-DD (i.e., the cases in which there is a feasible ideal point).

By Proposition 8, the Pareto-optimal set consists of very few solutions: in our tests we found a single non-dominated solution in 1278 out of 1500 cases, two in 210 cases and three in the remaining 6 cases. In 905 out of 1278 cases we proved that the Pareto-optimal set consists of a single solution. SVC-DD found the ideal point, and therefore certified optimality, in 590 cases, whereas a solution with $\bar{n} = N_{LB}$ bins was found in the remaining 315 cases.

Let us now describe the algorithm performance on the set $\mathcal{I}_R$ of real industrial instances. Table 9 in the Appendix distinguishes results by due-date groups A, B and C. Columns $|\bar{X}|$ and CPU report the number of non-dominated solutions in $\bar{X}$ and the CPU time spent, respectively. Table 10 in the Appendix reports the values of the solutions π_n and π_ℓ found. Real instances appear a bit more challenging than artificial ones: the mean values of $\bar{R}_1$ and G_A are respectively 0.11 and 7.38 for group A, 0.09 and 7.36 for group B, 0.09 and 8.86 for group C: more than twice the values observed for $\mathcal{I}_B$. In addition, the mean CPU time required by group A was 9.35 seconds, whereas group B and C needed 8.57 and 8.31 seconds respectively: more than five times that required for $\mathcal{I}_B$. In more detail, on some instances we may find that SVC-DD performs extremely well: very limited G_A and $\tilde{R}$ values were achieved not only on instances with small m (e.g. r_1, r_2 and r_5), but also on those of more substantial size (e.g. r_{15}, r_{18}, r_{22} and r_{26}), with gaps G_A below 3% and $\tilde{R}$ at most 0.02 for all groups. On the other hand, the algorithm struggles on other instances. Indeed, on r_3 and r_{14} the measure G_A peaks up to 40.73%, and $\tilde{R}$ touches 0.60; on r_4, r_{10} and r_{13} (without being exhaustive) the values of G_A and $\tilde{R}$ are significant, ranging between 10% and 25% for the former and moving from 0.14 and 0.22 for

the latter. As for $\mathcal{I}_B$, the vast majority of the frontiers found by SVC-DD consists of a single point, but rather than a characteristic of optimal frontiers, in $\mathcal{I}_R$ this feature seems to be related to some inability of the algorithm in diversification. Indeed, we were able to certify a single-point optimal frontier in just 11 out of 28 cases of group A, and optimality in 4 cases only. The above feature is even more evident in the results of groups B and C: Proposition 8 proves a singleton optimal frontier (that is, $|\bar{X}| = 1$) in ten and seven cases respectively, whereas only in two and one single case the solution found was certified to be the ideal point. This scenario suggests, as future work, to try and improve SVC-DD by local search. An intuitive way to populate $\bar{X}$ can rely on exploring solutions with increasing N and decreasing $L_{\max}$: starting from a non-dominated packing, the search could anticipate the critical item for which $L_{\max} = \ell$, while enforcing a controlled delay on items with strictly positive $\tau\Delta_i(\ell)$, see §2.2.

As a final remark, let us provide a quick insight on the behaviour of SVC-DD for the fixed-orientation problem 2OBP-DD. In order to compute a N_{LB} suitable for 2OBP-DD, we make use of the bound L_{CCM} recalled in [8]. The features of $\bar{X}$ do not change substantially on $\mathcal{I}_B$-instances: SVC-DD found a single non-dominated solution in 1241 cases, two in 250 cases, three in 8 cases and four in a single case. In 934 cases of 1241 only a single non-dominated solution exists in the Pareto-frontier: for 548 instances our algorithm returned the ideal point, whereas in the other 386 cases the lower bound on the number of employed bins was reached. On the other hand, the mean values of $\bar{R}_1$ and G_A (referred to the same ideal point as 2RBP-DD) more than double in all the due-date groups A,B and C, while global mean CPU time decreases by almost 36% (from 1.46 to 0.94 sec).

On $\mathcal{I}_R$-instances, the differences between oriented and non-oriented results are less evident than in $\mathcal{I}_B$. In group A, $\bar{R}$ and G_A are on average 0.11 and 7.96, respectively: very close to the 2RBP-DD case; for both groups B and C, $\bar{R}$ reaches on average 0.12, whereas the mean G_A increases up to 10.01 (group B) and 11.76 (group C). Concerning the structure of Pareto-frontiers (respectively for groups A, B and C), a singleton optimal frontier was proved in 13, 11 and 10 cases and global optimum was found in 5, 4 and 2 instances. The complete set of results for 2OBP-DD is available from the corresponding author.

7. Conclusions

We considered a bi-objective extension of an orthogonal two-dimensional bin packing problem, where items are associated with due-dates, and we wish to minimize both the maximum lateness of the items and the number of bins required to pack them all. We discussed some basic properties of non-dominated solutions and their dependence by the packing time τ. Moreover, following the definition of approximation ratios defined for multi-criteria problem in [12], we showed how approximation algorithms for 2BP provide approximation results also for 2BP-DD.

To solve the problem in practice, we proposed a sequential value correction heuristic (SVC-DD) and used a large set of benchmark instances to compare its performance with MXGA of [6] and CPMIP of [21]. Results show that SVC-DD largely outperforms both algorithms, achieving in general better primal-dual gaps in a much smaller CPU time. We further tested SVC-DD on a set of new and more challenging instances derived from real-world orders: our heuristic obtained very good results in many instances and struggled in very few ones, always providing solutions in very reasonable computational time. Finally, we analyzed our results under a multi-objective perspective and gave details about the structure of the heuristic frontier built by SVC-DD, inferring also some features of the Pareto-optimal sets.

Acknowledgements

We are grateful to Julia A. Bennell that kindly provided us the instances used in [6] and to SCM for making available the industrial instances. We also wish to thank two anonymous Reviewers whose sharp remarks helped us improve the presentation of our results. This research was supported by the Italian Ministry of Education, National Research Program (PRIN) 2015, contract n. 20153TXRX9.

[1] Alves, C., F. Clautiaux, J. Carvalho and J. Rietz, *Dual-Feasible Functions for Integer Programming and Combinatorial Optimization: Basics, Extensions and Applications*, Springer International Publishing (2016); ISBN: 978-3-319-27602-1

[2] Arbib, C., and F. Marinelli, *Maximum lateness minimization in one-dimensional bin packing*, Omega Int. J. of Management Science (2016); DOI: 10.1016/j.omega.2016.06.003

[3] Arbib, C., and F. Marinelli, *On cutting stock with due dates*, Omega Int. J. of Management Science **46** (2014) 11-20

[4] Belov G., and G. Scheithauer, *Setup and open-stack minimization in one-dimensional stock cutting*, INFORMS Journal on Computing **19**, 1 (2007) 27-35

[5] Belov, G., G. Scheithauer and E.A. Mukhacheva, *One-dimensional heuristics adapted for two-dimensional rectangular strip packing*, J. of the Operational Research Society **59**, 6 (2008) 823-832

[6] Bennell, J.A., L-S. Lee, and C.N. Potts, *A genetic algorithm for two-dimensional bin packing with due dates*, Int. J. of Production Economics **145**, 2 (2013) 547-560

[7] Berkey, J.O., and P.Y. Wang, *Two-dimensional finite bin-packing algorithms*, J. of the Operational Research Society **38** (1987) 423-429

[8] Clautiaux, F., A. Jouglet, and J. El Hayek, *A new lower bound for the non-oriented two-dimensional bin-packing problem*, Operations Research Letters **35** (2007) 365-373.

[9] Crainic, T. G., G. Perboli, W. Rei and R. Tadei, *Efficient lower bounds and heuristics for the variable cost and size bin packing problem*, Computers & Operations Research, **38**, 11 (2011) 1474-1482.

[10] Cui, Y. P., Y. Cui and T. Tang, *Sequential heuristic for the two-dimensional bin-packing problem*, European Journal of Operational Research, **240**, 1 (2015) 43-53

[11] Detti, P., A. Agnetis and G. Ciaschetti, *Polynomial algorithms for a two-class multiprocessor scheduling problem in mobile telecommunications systems*, Journal of Scheduling **8**, 3 (2005) 255-273

[12] Ehrgott M., *Approximation algorithms for combinatorial multicriteria optimization problems*, International Transactions in Operational Research **7** (2000) 5-31

[13] Ehrgott, M., and M.M. Wiecek, *Multiobjective Programming*. In: Multiple Criteria Decision Analysis: State of the Art Surveys, (2005) 667-708, Springer, New York

[14] Harren, R., and R. van Stee, *Packing Rectangles into 2OPT Bins Using Rotations*. In: Gudmundsson J. (eds) Algorithm Theory – SWAT 2008. Lecture Notes in Computer Science, vol **5124** (2008), Springer, Berlin, Heidelberg

[15] Jansen, K., L. Prädel and U.M. Schwarz, *Two for One: Tight Approximation of 2D Bin Packing*. In: Dehne F., Gavrilova M., Sack JR., Tóth C.D. (eds) Algorithms and Data Structures. WADS 2009. Lecture Notes in Computer Science, vol **5664** (2009), Springer, Berlin, Heidelberg

[16] Lodi, A., S. Martello, M. Monaci, C. Cicconetti, L. Lenzini, E. Mingozzi, C. Eklund and J. Moilanen, *Efficient two-dimensional packing algorithms for mobile WiMAX*, Management Science **57**, 12 (2011) 2130-2144

[17] Lodi, A., S. Martello, and D. Vigo, *Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems*, INFORMS Journal on Computing **11** (1999b) 345-357

[18] Lodi, A., S. Martello, and D. Vigo, *Recent advances on two-dimensional bin packing problems*, Discrete Applied Mathematics **123** (2002) 379-396

[19] Marinelli, F., and A. Pizzuti, *A Sequential Value Correction heuristic for a bi-objective two-dimensional bin-packing*, Electronic Notes in Discrete Mathematics **64** (2018) 25-34

[20] Papadimitriou, C.H., and M. Yannakakis, On the approximability of trade-offs and optimal access of web sources, *Proceedings 41st Annual Symposium on Foundations of Computer Science* (2000) 86-92

[21] Polyakovskiy, S., and R. M'Hallah, A hybrid feasibility constraints-guided search to the two-dimensional bin packing problem with due dates, *European J. of Operational Research* **266**, 3 (2018) 819-839

[22] Qi, X., A note on worst-case performance of heuristics for maintenance scheduling problems, *Discrete Applied Mathematics* **155** (2007) 416-422

[23] Reinertsen, H., and T.W.M. Vossen, The one-dimensional cutting stock problem with due-dates, *European J. of Operational Research* **201** (2010) 701-711

[24] SCM Group. `https://www.scmgroup.com/en_US`

[25] T'kind, V., and J.C. Billaut, *Multicriteria Scheduling Theory, Models and Algorithms*, (2006), Springer, Berlin, Heidelberg

[26] Uzsoy, R., *Scheduling a single batch processing machine with non-identical job sizes*, International Journal of Production Research **32** (1994) 1615-1635

[27] Wäscher, G., H. Haußner, and H. Schumann, *An improved typology of cutting and packing problems*, European J. of Operational Research **183** (2007) 1109-1130

[28] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., and V.G. da Fonseca, *Performance assessment of multiobjective optimizers: an analysis and review*, IEEE Transactions on Evolutionary Computation, **7** (2003) 117-132