Omega ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Omega

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

Maximum lateness minimization in one-dimensional bin packing ☆

Claudio Arbib [a], Fabrizio Marinelli [b,*]

[a] Dipartimento di Scienze/Ingegneria dell'Informazione e Matematica, Università degli Studi dell'Aquila, Via Vetoio, Coppito, I-67010 L'Aquila, Italy
[b] Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, Via Brecce Bianche, I-60131 Ancona, Italy

ARTICLE INFO

Article history:
Received 17 June 2015
Accepted 8 June 2016

Keywords:
One-dimensional bin packing
Scheduling
Mixed Integer programming
Integer reformulation

ABSTRACT

In the One-dimensional Bin Packing problem (1-BP) items of different lengths must be assigned to a minimum number of bins of unit length. Regarding each item as a job that requires unit time and some resource amount, and each bin as the total (discrete) resource available per time unit, the 1-BP objective is the minimization of the makespan $C_{max} = \max_j\{C_j\}$. We here generalize the problem to the case in which each item j is due by some date d_j: our objective is to minimize a convex combination of C_{max} and $L_{max} = \max_j\{C_j - d_j\}$. For this problem we propose a time-indexed Mixed Integer Linear Programming formulation. The formulation can be decomposed and solved by column generation relegating single-bin packing to a pricing problem to be solved dynamically. We use bounds to (individual terms of) the objective function to address the oddity of activation constraints. In this way, we get very good gaps for instances that are considered difficult for the 1-BP.

1. Introduction

In Bin Packing, a set J of n items of distinct sizes must be assigned to a minimum number of identical bins, so that the size of the items assigned to any bin never exceed its capacity. In the (orthogonal) s-dimensional problem, items and bins are closed intervals of IR^s, and the former must be placed into the latter with no overlap. Items can or cannot be rotated before placement: in the latter case, the edge lengths of each interval can be normalized, and bins become unit s-cubes.

One can interpret the s-dimensional Bin Packing as a scheduling problem with n jobs of unit time length: when scheduled, job j consumes some fraction of a discretized resource, the bin, available in one unit per time unit. In general, applications include all those cases (e.g., ads scheduling in sponsored internet search [1]) in which the resource used has both a geometric and a time dimension. Here are other popular applications:

- in s-dimensional cutting, jobs are parts to be produced, and the resource is a stock of standard size from which smaller items must be cut [2, 5–7,14,17, 21, 31];
- in telecommunication channel scheduling, jobs are packets of known length, and the resource is a frame able to host packets up to a given total length [4,11].

Under common assumptions, completion times corresponds to stock positions in the sequence, and minimizing C_{max} means minimizing the number of resource units used: standard sizes in cutting problems, frames in packet scheduling, etc. But C_{max} is just one of the many scheduling objectives one can be interested in. To generalize, call C_j the completion time of j (that is: item j is assigned to the C_j-th bin) and associate j with a cost function $f_j(C_j)$. In multi-objective scheduling, a solution is evaluated through several functions $f_j^k(C_j)$, $k = 1, \ldots, R$. Often, a multi-objective is summarized by a convex combination of functions obtained from the $f_j^k(C_j)$:

$$f(C_1, \ldots, C_n) = \sum_{k=1}^{r} \alpha_k \max_{j \in J}\{f_j^k(C_j)\} + \sum_{k=r+1}^{R} \alpha_k \sum_{j \in J} f_j^k(C_j)$$

with $\sum_{k=1}^{R} \alpha_k = 1, \alpha_k \geq 0$, $k = 1, \ldots, R$.

If a function is non-decreasing with C_j, then it is called *regular* [20, chapter 2]. When item j is due by a specific date d_j, the following regular functions are frequently taken into consideration:

- *Tardiness*: $f_j(C_j) = T_j = \max\{C_j - d_j, 0\}$;
- *Lateness*: $f_j(C_j) = L_j = T_j - E_j = C_j - d_j$.

☆This manuscript was processed by Associate Editor Kis.

* Corresponding author.
E-mail addresses: claudio.arbib@univaq.it (C. Arbib), fabrizio.marinelli@univpm.it (F. Marinelli).

http://dx.doi.org/10.1016/j.omega.2016.06.003
0305-0483/

1.1. Our problem

The general function $f(C_1,\ldots,C_n)$ combines min-max and min-sum terms. In this paper we focus on a pure min–max problem with $r=R=2$:

$$f_j^1(C_j)=C_j \quad f_j^2(C_j)=C_j-d_j$$

The second function is the lateness of item j: namely, we seek for a pack-and-schedule that solves

$$\min_{C_1,\ldots,C_n} f(C_1,\ldots,C_n)=\alpha_1 C_{max}+\alpha_2 L_{max} \tag{1}$$

for given rational constants $\alpha_1,\alpha_2\geq 0$ such that $\alpha_1+\alpha_2=1$. The opportunity of giving different weights to material and lateness costs is much application-dependent. There are relevant industrial cases in which material cost is closely comparable to, and sometimes larger than, the cost of delay (see e.g. [3]). In any case, α_1 and α_2 derive from the real costs of bin usage and time. When these costs cannot be easily evaluated, it is more appropriate to keep separate goals and transform one or both terms of the objective function into constraints (if both, we face a feasibility problem and speak of *goal-programming*). Our model naturally fits with this approach, see Section 2.4.

Models and methods will be developed according to the following

Assumption 1.1. In the definition of problem (1), we assume:

(i) *constant cut time* (this common assumption, see [2,7,21,26,31], may however be not obvious, especially for $s>1$: as item dimension increases, the time for item placement in – or cut from – a bin may change very much from pattern to pattern);
(ii) *due-dates integer multiple of cut time* (irrelevant for other cost functions, such as tardy jobs, but generally not irrelevant for lateness).

1.2. Literature review

Scheduling objectives in cutting and packing problems are receiving increasing attention. Among many papers concerned on cutting (see bibliography), [2,7,21] are the most recent and the closest to our situation:

- Reference [21] proposes an integer programming based heuristic to minimize a combination of trim-loss ($=C_{max}$) and total weighted tardiness.
- Reference [2] addresses the same problem as [21] by exact models, either with or without column generation. The model has variables associated with time-periods: in order to limit the number of variables, period lengths are adjusted by an ad-hoc procedure.
- Reference [7] develops a genetic heuristic for 2-dimensional, non-oriented, single bin size trying to approximate the Pareto frontier for the criteria of bin and maximum lateness minimization: this is the problem considered by us, although our computational experience is limited to 1-dimensional packing.

Parallel machine scheduling is a classical counterpart of bin packing: instead of being minimized, bins are given and the typical objective is to minimize the makespan (intended as the maximum load of a bin). Indeed, bin packing and parallel machines scheduling can be seen as "orthogonal" special cases of Cumulative Resource Scheduling [16], a problem in which each job consumes some amount of a shared resource up to availability.

A more general additive criterion is considered in [9,27–29], where precedence or time-indexed formulations are developed and decomposed in order to solve the problem by column generation:

- Reference [9] formulates $P\|\sum f_j(C_j)$ using decision variables that describe precedence relations among jobs on any machine; the master problem derived from reformulation is in the shape of Set Partitioning.
- Reference [27] focuses on $P\|\sum w_jC_j$ and directly formulates the decomposed problem with variables associated to feasible machine schedules. See also [28].
- Reference [29] assumes the general criterion $\sum f_j(C_j)$, and – as in our case – decomposition is applied to a time-indexed model. Unlike our case, however, jobs have non-unit processing times and do not consume other resource but time: the problem has therefore a special structure (interval matrix).

The time-indexed approaches listed above are very close to ours: the main difference is that we do not deal with a parallel-machine setting, therefore Dantzig–Wolfe decomposition is applied to different formulations.

A final note on complexity: $P\|(\alpha_1 C_{max}+\alpha_2 L_{max})$ is NP-hard for two machines and any values of $\alpha_1,\alpha_2\geq 0$ with $\alpha_1+\alpha_2=1$; when due dates are identical, the problem becomes Multiprocessor Scheduling. On the other hand, $1\|C_{max}$ is trivial and $1\|L_{max}$ can be solved in $n\log(n)$ time by Early Due Date priority rule (EDD). Whatever are the due dates, a schedule minimizing L_{max} is necessarily *active* [20, ch. 2], hence its C_{max} always equals the sum of processing times. Thus, $1\|(\alpha_1 C_{max}+\alpha_2 L_{max})$ has the same optimal solution as $1\|L_{max}$. For a comprehensive discussion on multicriteria scheduling problems see [25].

1.3. This contribution

Since bin packing is equivalent to cutting stock with unit demand, one can tackle due dates by a formulation of the cutting stock problem as in [2]. The general method is close to that applied to parallel scheduling in [27,28]. The objective here considered replaces however total tardiness with maximum lateness (see also [7]): with time-indexed formulations a min-sum term can in fact be less problematic because, unlike min-max, does not need activation constraints.

In this paper we show how to obtain guaranteed approximation algorithms for this problem from existing guaranteed approximation algorithms for Bin Packing. We also propose exact Mixed Integer Linear Programs based on time-indexing, and improve them by a careful use of lower and upper bounds so as to solve difficult problem instances. The main dataset for the numerical experiments was constructed on non-IRUP bin packing problems from the literature: with our approach, we were able to solve in few seconds problems with up to 120 parts and, for problems with 200 parts, reach very small gaps (less than 2%) in acceptable time (less than 600 s).

Formulations and an approximation result are detailed in Section 2. We investigate ways to improve the formulation so as to address quite large problem instances: a key issue to achieve efficiency is to take advantage of lower and upper bounds to C_{max}, L_{max} and to the global objective function (1). The bounds and their implementation in the formulations are discussed in Section 3. A computational experience based on [12] and mainly focussed on $\alpha_1=\alpha_2$ proves the validity of the method, and is reported in Section 4. Conclusions and directions for future research are drawn in Section 5.

Please cite this article as: Arbib C, Marinelli F. Maximum lateness minimization in one-dimensional bin packing. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.06.003

2. Notation, formulation and problem properties

After introducing the notation used throughout the paper (Section 2.1) we formulate the problem as Mixed Integer Linear Programming (MILP, Section 2.2); we then briefly survey some basic properties of the problem (Section 2.3), and finally pass to describe a MILP reformulation (Section 2.4).

2.1. Notation

From now on, we will assume items ordered by earliest due date (EDD): $0 \le d_1 \le \ldots \le d_n$.

We will denote as $z^{\mathcal{A}} = \alpha_1 C^{\mathcal{A}}_{max} + \alpha_2 L^{\mathcal{A}}_{max}$ the value of the solution obtained by a generic (either heuristic or optimal) algorithm $\mathcal{A}$ designed to solve (1). The notation $z^* = \alpha_1 C^*_{max} + \alpha_2 L^*_{max}$ will be reserved to optimal solutions, i.e. those minimizing (1). Because the same value of z^* can be attained by different pairs (C^*_{max}, L^*_{max}), the notation will in the following refer to such pairs and not to individual values of maximum completion time and lateness.

To help the reader, Table 1 collects the symbols used throughout the paper for objective function values, individual terms of the objective function, and bounds.

2.2. Formulation

Let $J = \{1, \ldots, n\}$ denote the item (or job) set and $T = \{1, 2, \ldots\}$ be a discrete planning horizon with m unit slots (we can initially assume $m = n$). Suppose that a 1-dimensional bin is made available in unit amount at each time slot, and let $0 < a_j \le 1$ denote the amount of bin used by job j upon completion. The discrete nature of the resource means that leftovers at time t cannot be used at any time $\neq t$. Hence, minimizing the total amount of resource used corresponds to minimize the time $C_{max} = \max_j\{C_j\}$ required to complete all the jobs. In addition, we assume that each item j is associated with a due date d_j, and define L_{max} as in Section 1.

For each $j \in J$ and $t \in T$, introduce

- a 0–1 variable x_{jt} that gets value 1 if and only if item j is assigned to the t-th bin;
- a 0–1 variable z_t that gets value 1 if and only if a bin is used at time t.

With this position, we formulate the problem as:

$$(\text{KX}) \quad \min \alpha_1 C_{max} + \alpha_2 L_{max}$$

$$\sum_{t \in T} x_{jt} = 1 \quad j \in J \tag{2}$$

$$\sum_{j \in J} a_j x_{jt} - z_t \le 0 \quad t \in T \tag{3}$$

$$\sum_{t \in T} z_t - C_{max} = 0 \tag{4}$$

$$t x_{jt} - L_{max} \le d_j \quad j \in J, t \in T \tag{5}$$

$$x_{jt}, z_t \in \{0, 1\} \quad j \in J, t \in T \tag{6}$$

Equality (2) ensures that every job is completed; inequality (3) that the resource is consumed according to availability; inequalities (4) and (5) define the terms of objective function (1). The notation (KX) indicates that the formulation extends, by time-indexing, the classical "Kantorovich" formulation (K) of Bin Packing (see [22]).

2.3. Basic problem features

Problem (1)–(6) is NP-hard even with $\alpha_1 = \alpha_2$ and $d_j = 0$ for all j, or when one of the two terms is neglected. For identical due dates and $\alpha_1 = 0$, we obtain the ordinary BinPacking.

The minimization of C_{max} is indeed related to that of L_{max}. In particular, any ϵ-approximate heuristic for the traditional bin packing problem (e.g., [23,24]) is also able to approximate the optimum of (1):

Proposition 2.1. *Let $L^*_{max} \ge 0$ and $z^{\mathcal{H}}$ be the value of a solution obtained by an ϵ-approximated algorithm $\mathcal{H}$ for 1-BP. Suppose also $\alpha_1 > 0$. Then*

$$\frac{z^{\mathcal{H}}}{z^*} \le \frac{1+\epsilon}{\alpha_1}$$

Proof. Let $C^{\mathcal{B}}$ be the minimum number of bins necessary to accommodate all items. Since $\mathcal{H}$ is ϵ-approximated,

$$z^{\mathcal{H}} = \alpha_1 C^{\mathcal{H}}_{max} + \alpha_2 L^{\mathcal{H}}_{max} \le \alpha_1 (1+\epsilon) C^{\mathcal{B}} + \alpha_2 L^{\mathcal{H}}_{max} \le (1+\epsilon)(\alpha_1 + \alpha_2) C^{\mathcal{B}}$$
$$= (1+\epsilon) C^{\mathcal{B}}$$

(the second inequality holds because the maximum lateness of $\mathcal{H}$ cannot exceed $C^{\mathcal{H}}_{max}$). If $L^*_{max} \ge 0$, then $z^* \ge \alpha_1 C^*_{max} \ge \alpha_1 C^{\mathcal{B}}$ (in fact $\mathcal{B}$ finds an optimum to Bin Packing). For $\alpha_1 > 0$, the thesis is then readily obtained. □

Corollary 2.2. *First Fit Decreasing and Best Fit Decreasing [24] approximate (1) within a ratio $\frac{3}{2\alpha_1}$.*

C_{max} and L_{max} may be conflicting objectives, though:

Table 1
Notation for (bounds to) objective function, bin number and max lateness.

Symbols	Description	Objective
$C^*_{max}, L^*_{max}, z^*$	Values of an optimal solution	(1)
C_{max}, L_{max}, z	Values of a feasible solution	–
$C^{\mathcal{A}}_{max}, L^{\mathcal{A}}_{max}, z^{\mathcal{A}}$	Values of the feasible solution returned by some algorithm $\mathcal{A}$	–
$C^{\mathcal{B}}_{max}, L^{\mathcal{B}}_{max}, z^{\mathcal{B}}$	Values of a solution minimizing the number of bins	$\min C_{max}$
$C^{\mathcal{H}}_{max}, L^{\mathcal{H}}_{max}, z^{\mathcal{H}}$	Values of a solution approximating the minimum number of bins	$\min C_{max}$
$C^{\mathcal{E}}_{max}, L^{\mathcal{E}}_{max}, z^{\mathcal{E}}$	Values of a solution minimizing the maximum lateness (EDD)	$\min L_{max}$
C_j, L_j	Completion time and lateness of item j in a generic solution	–
C_{LB}, L_{LB}, z_{LB}	Generic lower bounds to C_{max}, L_{max}, z, (Section 3.1)	–
$C_{LB} = C^{\phi}$	Lower bound to C_{max} derived by generic dual feasible function(s) ϕ	–
$C_{LB} = C^{\mathcal{G}}$	Lower bound to C_{max} by Gilmore-Gomory model	–
$C_{LB} = C^{\mathcal{CCM}}$	Lower bound to C_{max} by Carlier's et al. DFF [8]	–
$C_{LB} = C^{\mathcal{MT}}$	Lower bound to C_{max} by Martello-Toth DFF [19]	–
$L_{LB} = L^{\mathcal{R}}$	Lower bound to L_{max} by Proposition 3.2	–
$L_{LB} = L^{\mathcal{ER}}$	Best bound between $L^{\mathcal{E}}_{max}$ and $L^{\mathcal{R}}$	–
$z_{LB} = z^{\mathcal{GER}}_{LB}$	Lower bound to z combining $C^{\mathcal{G}}$ and $L^{\mathcal{ER}}$	–
$z_{LB} = z^{\text{HX}}_{LB}, z^{\text{GX}}_{LB}, z^{\text{GX}'}_{LB}$	Lower bound to z by continuous relaxation of models (HX), (GX), (GX′)	–
C_{UB}, L_{UB}, z_{UB}	Generic upper bounds to C_{max}, L_{max}, z (Section 3.2)	–
$z_{UB} = z^{\text{S\&F}}_{UB}$	Upper bound to z by sort-and-fit heuristic	–
$z_{UB} = z^{\text{SVC}}_{UB}$	Upper bound to z by Sequential Value Correction heuristic	–
$z_{UB} = z^{\text{PB}}_{UB}$	Upper bound to z by Price-and-Branch Algorithm	–

Please cite this article as: Arbib C, Marinelli F. Maximum lateness minimization in one-dimensional bin packing. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.06.003

Example 2.3. Take $n=2h$, item sizes

$$a_j=\frac{2j}{h(h-1)} \quad a_{h+j}=1-a_j$$

and due dates $d_j=1, d_{h+j}=h+1$ for $j=1,\ldots,h$.

A solution minimizing C_{max} has exactly h bins of the form $\{j, h+j\}$. In this solution, $L_{max}=L_h=h-1$. A solution minimizing L_{max} has instead $h+1$ bins, the first of which is $\{1,\ldots,h\}$. Here, $L_{max}=0$ and $C_{max}=h+1$. Thus, minimizing C_{max} may not help minimizing L_{max}, and vice-versa. □

Suppose to use an algorithm $\mathcal{B}$ that finds the minimum C_{max} as a heuristic for minimizing (1): that is, $\mathcal{B}$ approximates the minimum number of bins within a ratio $\epsilon=0$. With $\alpha_1=\alpha_2=\frac{1}{2}$ in Example 2.3, the ratio between the heuristic and the optimum value is

$$\frac{\alpha_1 h+\alpha_2(h-1)}{\alpha_1(h+1)+\alpha_2\cdot 0}=\frac{2h-1}{h+1}$$

which tends to 2 when $h\to\infty$. Thus Example 2.3 shows that the bound given by Proposition 2.1 is asymptotically tight. The bound clearly worsens as α_1 decreases and eventually diverges for $\alpha_1=0$. This behavior is however unavoidable even for heuristics specifically designed for minimizing L_{max}, since L^*_{max} could be zero.

2.4. Reformulation

Let $\mathbf{y}_t^p\in\{0,1\}^n$, $p\in P\cup\{\mathbf{0}\}$, be the incidence vectors of job sets fulfilling knapsack constraint (3): such vectors are called *patterns*. For any $t\in T$, rewrite an integer solution $\mathbf{x}_t=(x_{1t},\ldots,x_{nt})$ of (3) as

$$\mathbf{x}_t=\sum_{p\in P}\lambda_t^p\mathbf{y}_t^p \quad \sum_{p\in P}\lambda_t^p\le 1 \quad \lambda_t^p\ge 0 \text{ and integer } \forall p\in P$$

The first position, written for $j\in J$, becomes

$$x_{jt}=\sum_{p\in P_j}\lambda_t^p$$

where P_j denotes the set of patterns p that *cover* j, i.e., those such that $y_j^p=1$. Replacing in (2)–(6) we get the following formulation (GX), that can be seen as a time-indexed extension of the well-known pattern-based formulation (G) of Bin Packing (see [15]):

$$\text{(GX)}\quad \min\ \alpha_1 C_{max}+\alpha_2 L_{max}=\alpha_1\sum_{t\in T}\sum_{p\in P}\lambda_t^p+\alpha_2 L_{max}$$

$$\sum_{t\in T}\sum_{p\in P_j}\lambda_t^p=1 \quad j\in J$$

$$\sum_{p\in P}\lambda_t^p\le 1 \quad t\in T$$

$$f_j^2(t)\sum_{p\in P_j}\lambda_t^p-L_{max}\le 0 \quad j\in J,\ t\in T$$

$$\lambda_t^p\ge 0 \text{ and integer} \quad p\in P,\ t\in T$$

where $f_j^2(t)=L_j=t-d_j$ (note that (GX) can easily be generalized to other objectives: it suffices re-define $f_j^2(t)$). Variables λ_t^p get value 1 if pattern p is adopted at time t, and 0 otherwise: therefore their sum over P and T gives the total amount of bins used, that is C_{max}. In (GX), patterns are defined offline according to any resource type and consumption scheme. Thus, (GX) is independent on the resource dimension s, that instead contributes to define the pricing problem to be solved for pattern generation. Unlike the Dantzig–Wolfe decomposition of ordinary cutting stock or bin packing, the generation of a promising pattern must be iterated for all the $t\in T$. For $s=1$, each pattern generation is a 0–1 Knapsack.

The Pareto frontier of C_{max} vs. L_{max} is usually explored by separately constraining the terms the objective function consists of. Model (GX) fits very efficiently to this purpose. Specifically, the number of bins can be constrained via the time horizon T; the lateness by excluding from the t-th pricing problem all the items that have, at t, a lateness larger than a given upper bound L_{UB}. Observe that reasonable trade-off solutions spread over a small interval $[C_{LB}, C_{UB}]$ of used bins. Therefore, Pareto-optimal solutions are found by solving with $\alpha_1=0$ the few problems for $|T|\in[C_{LB}, C_{UB}]$, or equivalently with $\alpha_1=1$ the few problems with $L_{max}\le L_{UB}$.

3. Use of bounds

A careful use of bounds in formulations (KX) and (GX) definitely improves their performance. Dual (i.e., lower) and primal (i.e., upper) bounds to the objective function play a well-known role in pruning the search tree during branch-and-bound and reducing optimality gaps. But separate bounds to the terms that form the objective function are also very useful. On the one hand, upper bounds to C_{max} can help

- reduce T and hence the number of time-indexed variables in both (KX) and (GX).

On the other hand, lower bounds to L_{max} can be fruitfully employed to

- fix variables x_{jt} of (KX) through constraints (5);
- improve the dual LP bound of (GX) through the reduction of the "big M" $f_j^2(t)$ in the third set of inequalities.

Fig. 1 shows a schematic drawing of the objective function space C_{max} vs. L_{max}. Appealing values belong to the dark grey triangle: the combination of bounds to the two terms and to the objective function as a whole allow to infer new (and possibly stronger) specific bounds to each term.

3.1. Lower bounds

If C_{LB}, L_{LB} are individual lower bounds to C_{max}, L_{max}, then

$$z_{LB}=\alpha_1\lceil C_{LB}\rceil+\alpha_2\lceil L_{LB}\rceil$$

is indeed a lower bound to (1) – notice that round up of L_{LB} is authorized by Assumption 1.1 (ii) (integer due dates). We can prove the following:

Proposition 3.1. *Let C_{LB} be any lower bound to C_{max}, and F^* be the value of a schedule that minimizes $F=\max\{f_j(C_j)\}$. If the f_j are*

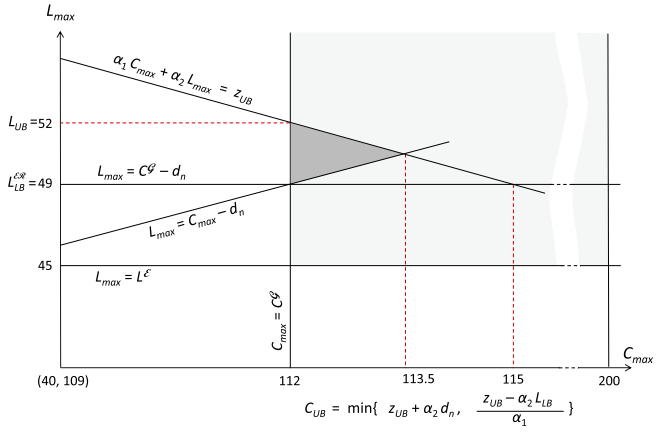

Fig. 1. Schematic drawing of the objective function space ($\alpha_1=0.5, z_{UB}=82, d_n=63$).

Please cite this article as: Arbib C, Marinelli F. Maximum lateness minimization in one-dimensional bin packing. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.06.003

regular, then

$\alpha_1\lceil C_{LB}\rceil+\alpha_2 F^*$

is a lower bound to $f=\alpha_1 C_{max}+\alpha_2 F$.

Proof. Under regularity, F^* is in fact a lower bound to the value of F attained by a minimizer of f, see [2]. □

Proposition 3.1 provides an easy way to compute a lower bound to (1). Sequence the items one after the other in EDD order: let $C^{\mathcal{E}}=\sum_{j\in J}a_j$ be the completion time of J, and $L^{\mathcal{E}}$ be the maximum lateness so obtained. Since EDD is a minimizer of L_{max}, Proposition 3.1 implies

$$z_{LB}^{\mathcal{E}}=\alpha_1\lceil C^{\mathcal{E}}\rceil+\alpha_2\lceil L^{\mathcal{E}}\rceil \tag{7}$$

Note that, although trivial, (7) improves the lower bound computed by solving the continuous relaxation (KX$_R$) of (KX): in fact, the C_{max} term of (KX$_R$) equals $C^{\mathcal{E}}$ and, unlike the EDD sequencing, (KX$_R$) does not ensure ≤ 2 item fractions per bin.

Let us examine a few ways of improving (7) by strengthening $C^{\mathcal{E}}$ and $L^{\mathcal{E}}$.

Lower bounds to C_{max} can in general be obtained via *Dual Feasible Functions* (DFF), see [10]:

Definition 3.1. A function $\phi:[0,1]\to[0,\phi(1)]$ is dual feasible if

$$\sum_{x\in S}x\leq 1\Rightarrow\sum_{x\in S}\phi(x)\leq\phi(1)$$

for any finite discrete $S\subseteq[0,1]$.

Definition 3.1 is used to introduce a lower bound $C^{\phi}=\lceil\sum_{j\in J}\phi(a_j)\rceil$ to C_{max}. A trivial DFF is $\phi(a_j)=a_j$, $\phi(1)=1$, that gives the lower bound $C^{\phi}=\lceil\sum_{j\in J}a_j\rceil=\lceil C^{\mathcal{E}}\rceil$ used in (7). Normally, a strict lower bound $C^{\mathcal{G}}$ to C_{max} is computed by the continuous relaxation of Gilmore–Gomory's model (G) for the Bin Packing. In fact, $C^{\mathcal{G}}$ is a DFF: if π_j^* is an optimal dual solution to (G), then $\phi(a_j)=\pi_j^*$ and $\phi(1)=1$ imply $C^{\phi}=\lceil C^{\mathcal{G}}\rceil\leq C_{max}$. One can prove that $C^{\mathcal{G}}$ is the best possible DFF. However, its computation is expensive, because the π_j^* cannot be obtained in closed form and require column generation. A trade-off between CPU time and bound quality is offered by the families of DFF proposed by Carlier et al. [8], and by those implicitly adopted by Martello and Toth [19]: the former experimentally proved to give the best trade-off among the functions most frequently used for Bin Packing, see [10]. Both the bounds derived, respectively denoted as $C^{\mathcal{CCM}}$ and $C^{\mathcal{MT}}$, can be computed in polynomial time. In our experiments, $\max\{C^{\mathcal{CCM}},C^{\mathcal{MT}}\}$ turned out to be as effective as $C^{\mathcal{G}}$ (see Section 4).

Lower bound $L^{\mathcal{E}}$ can be improved as follows:

Proposition 3.2. *Let* $m(k)$ *be any lower bound to the minimum number of bins necessary to accommodate items* $1,\ldots,k$. *Then*

$$L^{\mathcal{R}}=\max_{1\leq k\leq n}\left\{\max_{0\leq j<m(k)}\{m(k)-j-d_{k-j}\}\right\} \tag{8}$$

is a lower bound to L^*_{max}.

Proof. Call L^t the maximum lateness attained in bin t by a minimizer of (1). The first k items require no less than $m=m(k)$ bins, and each bin contains at least one element. Thus the minimizer assigns at least an item $j\in K=\{1,\ldots,k\}$ to a bin $t\geq m$. Since k is the less urgent of these items,

$$L^*_{max}\geq L^t\geq L_j=t-d_j\geq m-d_j\geq m-d_k \tag{9}$$

Suppose now that the minimizer places item $i\in K$ in bin $m-1$. If $i=k$, then k is not placed in a bin $t\geq m$, therefore inequality (9) becomes

$$L^*_{max}\geq L^t\geq m-d_{k-1}$$

If instead $i\neq k$, then

$$L^*_{max}\geq L^{m-1}\geq L_i=(m-1)-d_i\geq(m-1)-d_{k-1}$$

Taking the most optimistic of the two cases above

$$L^*_{max}\geq\min\{m-1-d_{k-1},m-d_{k-1}\}=m-1-d_{k-1} \tag{10}$$

To get (8) we then just need to repeat the argument for bins $m-2,m-3,\ldots$, take the strongest of the $m(k)$ inequalities obtained, and then choose again the best bound for $k=1,\ldots,n$. □

By Propositions 3.1 and 3.2 we derive

$$L^{\mathcal{ER}}=\max\{\lceil L^{\mathcal{E}}\rceil,L^{\mathcal{R}}\} \tag{11}$$

and

$$z_{LB}^{\mathcal{GER}}=\alpha_1\lceil C^{\mathcal{G}}\rceil+\alpha_2 L_{LB}^{\mathcal{ER}} \tag{12}$$

To shorten the computation of $L^{\mathcal{R}}$ in (11) one can first evaluate the inner argument of (8) by an easy bound $m(k)$ (e.g., $C^{\mathcal{E}}$ applied to the first k items), then focus on the best k and refine with, say $C^{\mathcal{G}}$.

Bound (11) can be used to strengthen model (GX). We can in fact write $L_{max}=L^{\mathcal{ER}}+\Delta L$, and reformulate the model using real variable ΔL:

$$\begin{aligned}
(\text{GX}')\quad & \alpha_2 L^{\mathcal{ER}}+\min\quad \alpha_1 C_{max}+\alpha_2\Delta L\\
& \sum_{t\in T}\sum_{p\in P_j}\lambda_t^p=1\quad j\in J\\
& \sum_{p\in P}\lambda_t^p\leq 1\quad t\in T\\
& \max\{0,t-d_j-L^{\mathcal{ER}}\}\sum_{p\in P_j}\lambda_t^p-\Delta L\leq 0\quad j\in J,\ \ t\in T\\
& \lambda_t^p\geq 0\text{ and integer}\quad p\in P,\ \ t\in T
\end{aligned}$$

so taking advantage of a reduced coefficient in the right-hand side of the third set of constraints.

3.2. Upper bounds

A simple upper bound to C_{max} is immediately obtained:

Proposition 3.3. *Let* $C^{\mathcal{B}}$ *be the minimum number of bins in a traditional* Bin Packing *problem. Then a minimizer of* (1) *uses* $<\min\{n,2C^{\mathcal{B}}\}$ *bins.*

Proof. See [2]. □

Proposition 3.3 give the number m of time-slots that it is sensible to consider in formulations (KX) and (GX).

Example 3.4. For example, take HBPP.20015009.02720 [12]: this instance has $n=200$ items, and $C^{\mathcal{B}}=112$. Thus m can be set to $\min\{200,224\}=200$, implying 40,000 0–1 variables in model (KX), and 200 patterns generated per pricing iteration in model (GX).

The bound of Proposition 3.3 can be improved as follows:

Proposition 3.5. *Let* z_{UB},L_{LB} *respectively be any upper bound to* (1) *and lower bound to the term* L_{max} *in* (1). *Then*

$$C_{max}\leq\min\left\{\left\lfloor\frac{z_{UB}-\alpha_2 L_{LB}}{\alpha_1}\right\rfloor,\lfloor z_{UB}+\alpha_2 d_n\rfloor\right\}=C_{UB} \tag{13}$$

Proof. Inequality $\alpha_1 C_{max}+\alpha_2 L_{max}\leq z_{UB}$ is combined with $L_{max}\geq L_{LB}$ to obtain the first argument within brackets. We then observe

Please cite this article as: Arbib C, Marinelli F. Maximum lateness minimization in one-dimensional bin packing. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.06.003

that the item placed in the last bin has due-date, at the latest, d_n. Hence $L_{max} \le C_{max} - d_n$, and using $\alpha_1 + \alpha_2 = 1$ we obtain the second argument. □

In a similar way we get a bound to L_{max}:

Proposition 3.6. *Let z_{UB}, C_{LB} respectively be any upper bound to (1) and lower bound to the term C_{max} in (1). Then*

$$L_{max} \le \left\lfloor \frac{z_{UB} - \alpha_1 C_{LB}}{\alpha_2} \right\rfloor = L_{UB} \tag{14}$$

Proof. Trivial. □

Bound (14) allows variable fixing:

- In formulation (KX), $x_{jt} := 0$ for $t \ge d_j + L_{UB}$ and any $j \in J$.
- In formulation (GX), $\lambda_t^p := 0$ for $t \ge d_j + L_{UB}$ and any $p \in P_j$.

The quality of bounds (13) and (14) depends on the heuristic upper bound z_{UB}. In this paper, we tested two heuristics: *Sort-and-fit* (S&F) and *Sequential Value Correction* (SVC).

A Sort-and-fit heuristic is a combination of a bin packing heuristic and a sorting rule applied to J:

Algorithm *Sort-and-fit*

1. Sort items according to sorting rule;
2. If all items have been placed, then output $z_{UB}^{S\&F}$ and stop; else, assign the next unplaced item to the bin with smallest residual capacity (still sufficient to accommodate the item);
3. If no such item can be found, open a new bin; go to step 2.

We tried Sort-and-fit in two variants with different sorting rules:

(a) Earliest Due Date first (EDD) rule;
(b) EDD plus First Fit Decreasing (FFD) whenever due dates are identical.

A SVC heuristic [6] associates a pseudo-price π_j with each $j \in J$, and generates N solutions by maximizing the total pseudo-price of the items included in each bin. Pseudo-prices are conveniently updated at run time.

Algorithm *Sequential Value Correction*

Initialize pseudo-prices for all the items of J;
For N times, repeat:

1. Fill next bin by maximizing the total price of the items included;
2. If not all items have been accommodated yet, then go to step 1;
3. Otherwise, update z_{UB}^{SVC} to the best solution generated so far; update pseudo-prices.

The best choice of pseudo-prices depends on application. We obtained good results with pseudo-prices that take into account (i) due-dates (high price to the most urgent part) and (ii) size (high price to the largest part). The pseudo-price of an item is further increased if the item is responsible of the maximum lateness or of large trim-loss in the current solution.

In detail, pseudo-prices are parametrized with positive real $\rho_1, \rho_2, \rho_3, \rho_4$. Initialization is as follows:

$$\pi_j := \frac{\rho_1}{d_j + 1} + a_j \tag{15}$$

With $\rho_1 = 1000$, pseudo-prices turn out to be initialized by non-decreasing due dates and, for identical due dates, by non-increasing size (in our tests, item sizes are > 0.001). After a new solution is obtained, pseudo-prices are updated for each bin K of the solution and item $j \in K$:

$$\pi_j := \pi_j \left(1 + \frac{1 - \sum_{i \in K} a_i}{\rho_2}\right) \tag{16}$$

Moreover, if the maximum lateness occurs at item h, price π_h is further updated:

$$\pi_h := \pi_h \left[1 + \rho \left(\frac{C_h}{d_h + 1}\right)^{\rho_4}\right] \tag{17}$$

where ρ is a random variable uniformly distributed in $[1, 1 + \rho_3]$. In this way, the pseudo-price is more than doubled: the increase is positively correlated to the current maximum lateness (using $\rho_4 < 1$ the increase is sub-linear).

Example 3.7. Let us go back to instance HBPP.20015009.02720 (Example 3.4). Computing (13) with $z_{UB} = z_{UB}^{SVC}$ and $L_{LB} = L^{\mathcal{ER}}$, we get $C_{UB} = 113$: the variables of (KX) are so reduced to $200 \cdot 113 = 22{,}600$, and the patterns of (GX) generated per pricing iteration to ≤ 113.

As a general observation, Sort-and-fit is very quick – in fact, it runs in $O(n \log n)$ time. Sequential Value Correction, instead, is not polynomial since it requires the repeated solution of 0–1 KNAPSACK problems, but in general it provides better solutions (see Section 4).

Both Sort-and-fit variants approximate the optimal solution within a ratio $1.75/\alpha_1$ (see Proposition 2.1) since they can be read as on-line First Fit algorithms for BIN PACKING (see [24]). This ratio is theoretically worse than that of First Fit Decreasing or Best Fit Decreasing heuristics (Corollary 2.2); however, the latter algorithms do not take due dates into account and are therefore outperformed in practice by Sort-and-fit.

Upper bounds can be further improved by Price-and-Branch. Indeed, models (GX) and (GX′) provide a means for exactly solving the problem by Branch-and-Price, i.e., column generation within implicit enumeration. Branch-and-Price algorithms for 1-BP were proposed by Vanderbeck [30] and de Carvalho [26]. Those schemes can easily be adapted to (GX′), since the variables associated with each time slot are substantially the same as in the traditional 1-dimensional BIN PACKING. The design of an efficient exact algorithm for (GX′), however, goes beyond the scope of this paper. Here we are mainly interested to reduce the primal-dual gap one can obtain via the bounds of Sections 3.1 and 3.2. To this aim, we implemented the following procedure:

Algorithm *Price-and-Branch*

1. Find $C^{\mathcal{G}}$ solving the continuous relaxation of (G) by column generation;
2. Compute C_{UB}, L_{UB} and L_{LB};
3. Set model (GX′) with C_{UB} and L_{LB}, and initialize the master problem with the best primal solution of the Sort-and-fit and the Sequential heuristic (Section 3.2);
4. Compute lower bound $z_{LB}^{GX'}$ solving the continuous relaxation of (GX′) by column generation;
5. Restore the integrality constraints of the (partial) formulation of (GX′) obtained at step 4, and get a primal bound z_{UB}^{PB} by solving the resulting MILP.

The time required by step 1 of *Price-and-Branch* can be reduced by replacing $C^{\mathcal{G}}$ with a DFF bound, such as $C^{\mathcal{CCM}}$ and $C^{\mathcal{MT}}$ described in Section 3.1.

Please cite this article as: Arbib C, Marinelli F. Maximum lateness minimization in one-dimensional bin packing. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.06.003

Table 2
Lower bounds.

Inst. sets		z_{LB}^{KX}		$z_{LB}^{\mathcal{E}}$	$z_{LB}^{\mathcal{GER}}$		$z_{LB}^{GX'}$	
n	inst. (#)	Value	CPU time (s)	impr. (%)	impr. (%)	CPU time (s)	impr. (%)	CPU time (s)
20	1	6.0	<0.005	16.7	–	<0.005	7.1	0.09
40	3	10.7	0.02	12.2	–	0.02	4.2	0.27
60	9	15.5	0.05	15.0	0.5	0.05	2.8	0.57
80	3	17.0	0.09	15.7	–	0.17	2.5	1.60
100	4	23.5	0.38	17.0	–	0.19	1.8	2.09
120	10	31.0	0.54	16.4	0.5	0.24	1.4	2.92
140	3	41.0	0.76	15.8	0.6	0.18	1.1	3.37
160	7	45.3	0.99	14.8	1.3	0.31	1.1	5.01
180	5	51.8	1.37	15.4	1.2	0.38	0.8	7.35
200	8	53.6	1.48	16.3	1.0	0.70	0.8	12.16

Table 3
Primal-dual gaps.

Inst. sets		Basic		Combinatorial		Reformulation			Price-and-Branch		
n	inst. (#)	opt. (#)	Gap (%)	Opt. (#)	Gap (%)	Opt. (#)	Gap (%)	CPU time (sec.)	Opt. (#)	Gap (%)	CPU time (sec.)
20	1	1	0.0	0	7.1	1	0.0	0.09	1	0.0	0.11
40	3	0	4.0	0	12.0	0	4.0	0.27	0	4.0	0.49
60	9	0	3.9	0	9.3	1	2.4	0.57	1	2.4	0.62
80	3	2	0.8	0	5.1	1	1.6	1.60	1	1.6	1.59
100	4	1	6.3	0	9.1	0	2.6	2.09	0	2.2	3.48
120	10	0	13.4	0	8.3	1	1.7	2.92	2	1.3	4.51
140	3	0	25.3	0	9.2	0	2.1	3.37	0	1.0	19.90
160	7	0	23.7	0	7.9	0	2.6	5.01	0	1.3	82.34
180	5	0	24.0	0	8.5	0	2.7	7.35	0	1.4	176.56
200	8	0	29.7	0	8.0	0	2.1	12.16	0	1.9	547.09

We finally observe that the upper bound z_{UB}^{PB} computed at step 5 can be reused in (13), (14) to further improve the bounds on C_{max} and L_{max} in an iterated application of Algorithm *Price-and-Branch*, see Section 4.2.

4. Computational experience

Computational tests were done with (KX) and (GX) setting $T = C_{UB}$ according to (13), and with the respective strengthened versions (KX′) and (GX′) where we took advantage of upper and lower bound to L_{max}.

The algorithms were coded in C++ and compiled with Microsoft *cl* compiler (version 12.00.8804) with option /O2. Numerical precision was set to 10^{-6}. Problems were solved on an Intel® Xenon® E5620 2.40 GHz with 16Gb RAM. (Integer) linear programs were solved by Cplex 12.5 with default setting, initialized with the best primal solution available.

Due-dates were added to fifty-three non-IRUP bin packing benchmark instances with $n \in [20, 200]$, see [12], normalized as usual to unit stock lengths. Due-dates were randomly chosen in the interval $[0, 0.6\lfloor C^{\mathcal{G}} \rfloor]$, where $C^{\mathcal{G}}$ denotes as usual the optimum of the continuous relaxation of (G). All the instances tested can be downloaded from [18].

For all problems:

- we chose $\alpha_1 = \alpha_2 = 0.5$ in the objective function (1);
- the parameters of the SVC heuristic were set to $N = \rho_1 = 1000$, $\rho_2 = 100, \rho_3 = 0.05$ and $\rho_4 = 0.3$.

Tables 2–4 report the test results; rows corresponds to instance sets, and each entry is a mean value computed with reference to the number of instances in the set.

The experiments aimed to:

(i) Assess the quality of the dual bounds (7) and (12), and the tightness of model (GX′) (Section 3.1).
(ii) Compare the primal-dual gaps obtained by the basic model (KX), its reformulation (model (GX′) with SVC or Price-and-Branch), and the combinatorial approach (Sort-and-Fit heuristic vs. dual bound (7)) (Section 4.2).
(iii) Measure the effect on formulations of the bounds to L_{max} (Section 4.3).

4.1. Dual bounds

Table 2 shows how the combinatorial and the reformulation approaches progressively strengthen the basic (and somewhat trivial) lower bound z_{LB}^{KX} obtained by solving the continuous relaxation of (KX). Indeed, the combinatorial bound $z_{LB}^{\mathcal{E}}$ (7) sensibly improves z_{LB}^{KX}, and the method is extremely fast (CPU time always less than 0.005 s). The enhanced bound $z_{LB}^{\mathcal{GER}}$ (12) slightly improves $z_{LB}^{\mathcal{E}}$ (see column 6 of Table 2); the CPU time increases (because we need to perform column generation), see column 7 of Table 2, but in the majority of cases stays well under the time required by the basic linear program. On our benchmark instances, however, the bound $C_{LB} = C^{\phi} = \max\{C^{\mathcal{CCM}}, C^{\mathcal{MT}}\}$ provided by DFF works as well as $C^{\mathcal{G}}$: in fact, the relevant bound $z_{LB}^{\phi\mathcal{ER}}$ is obtained with a CPU time comparable to that needed for computing $z_{LB}^{\mathcal{E}}$.

In turn, the reformulation bound improves $z_{LB}^{\mathcal{GER}}$ at the expense of increasing roughly tenfold the CPU time, (see column 8 of Table 2). The quickness with which $z_{LB}^{\mathcal{E}}$ and $z_{LB}^{\mathcal{GER}}$ are computed make them most suitable for implementation within enumeration schemes. However, when the bound is used to assess the quality of a heuristic, this feature loses importance in comparison to the

Please cite this article as: Arbib C, Marinelli F. Maximum lateness minimization in one-dimensional bin packing. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.06.003

Table 4
Benefit from improved L_{max} lower and upper bounds.

Inst. set		Program (KX)			Program (GX)			Price-and-Branch on (GX)	
n	inst. (#)	z_{LB}^{KX}	CPU time (s)	Gap 1 h (%)	z_{LB}^{GX}	Cols (#)	CPU time (s)	z_{UB}^{PB}	CPU time (s)
20	1	6.0	0.00	0.00	6.0	161.0	0.03	7.5	0.44
40	3	10.7	0.02	4.05	10.8	587.0	0.39	13.0	1.40
60	9	15.5	0.05	3.93	15.9	1065.8	1.49	18.9	26.15
80	3	17.0	0.09	0.83	17.0	1850.3	5.57	20.5	6.02
100	4	23.5	0.38	6.30	23.8	2351.8	9.72	28.6	11.43
120	10	31.0	0.54	13.44	31.5	2694.4	14.82	37.4	384.53
140	3	41.0	0.76	25.29	41.8	3004.7	19.88	49.2	456.97
160	7	45.3	0.99	23.66	46.4	3480.3	31.39	54.4	1001.67
180	5	51.8	1.37	23.98	52.7	4394.0	50.96	62.3	1001.39
200	8	53.6	1.48	29.71	54.5	5267.9	84.62	64.9	990.59
		Program (KX′)			Program (GX′)			Price-and-Branch on (GX′)	
20	1	7.0	0.02	0.00	7.5	102.0	0.02	7.5	0.02
40	3	12.0	0.02	5.62	12.5	311.3	0.11	13.0	0.24
60	9	17.9	0.02	3.71	18.4	472.9	0.24	18.9	0.09
80	3	19.7	0.02	1.63	20.2	1031.3	1.08	20.5	0.15
100	4	27.5	0.03	2.56	28.0	989.8	1.30	28.6	1.57
120	10	36.2	0.05	2.11	36.8	1119.3	1.81	37.3	1.81
140	3	47.8	0.06	3.22	48.3	1115.3	1.90	48.8	16.67
160	7	52.3	0.10	3.89	53.3	1312.4	3.11	54.0	77.63
180	5	60.5	0.12	2.80	61.1	1600.8	4.90	62.0	169.58
200	8	63.0	0.13	2.80	63.6	2080.1	9.10	64.8	535.63

bound quality, since the result is often crucial to close the gap and thus certify optimality.

4.2. Gaps

Primal-dual gaps (Table 3) are referred to the following trials:

- basic: gap obtained by Cplex after 1 hour of computation on (KX)
- combinatorial: $z_{UB}^{S\&F} - z_{LB}^{\mathcal{E}}$, where $z_{UB}^{S\&F} = \min\{z_{UB}^{S\&F(a)}, z_{UB}^{S\&F(b)}\}$
- reformulation: $z_{UB}^{SVC} - z_{LB}^{GX}$
- Price-and-Branch: $z_{UB}^{PB} - z_{LB}^{GX'}$

As observed at the end of Section 3.2, Sort-and-Fit heuristics are very fast. Variant (b) produces slightly better results than variant (a) (1.87% on average) – although (b) does not always dominates (a) (in 7 cases (a) provides better solutions). The SVC algorithm definitely dominates S&F (solutions are improved by 5.7% on average) and runs in 1.13 s on average. In particular, the solutions provided by SVC are near-optimal for the C_{max} term (on average they have 1.34 bins more than the lower bound $C^{\mathcal{G}}$), but show an L_{max} on average 3.77 time-periods longer than $L^{\mathcal{E}}$ (with a mean gap of 17.2%).

For each approach and instance group, Table 3 shows the number of optima found and the optimality gap. CPU time is indicated for reformulation and Price-and-Branch only: in the basic case, the time limit of 3600 s was always reached, with the exception of the four cases in which an optimum was found; in the combinatorial case, CPU time turned out to be always under 0.005 s.

The combinatorial approach does not always improve the basic, but is much faster (the basic approach takes one hour CPU) and the result seems scalable, i.e. quite insensible to instance size. Noticeable gap reductions are obtained by the reformulation within few CPU seconds (time roughly increases with the square of problem size). The further computational effort required by step 5 of Algorithm *Price-and-Branch* is apparently not much worthy: gap reduction can indeed be remarkable, up to half in the largest instances; but the number of optima found is basically the same, and moreover, CPU time fits an exponential behaviour.

The gap can be improved by plugging into the formulation the bound z_{UB}^{PB} obtained at step 5 of Algorithm *Price-and-Branch*, and iterate the algorithm from step 2. With this method, we observed z_{UB}^{PB} improving z_{UB}^{SVC} in 19 cases of 53. After the third iteration (in 17 cases, after the second), no improvement is obtained. The lower bound slightly increases but not enough to change after round-up. The upper bound decreases in 3 cases, in 2 of which the primal-dual gap is reduced to half. CPU time clearly grows much, but not the number of columns, that increases on average by some 1%.

4.3. Effect of improved bounds

The improved lower and upper bounds (11) and (14) to L_{max} not only have a strong effect on both (KX) and (GX), but also on the Price-and-Branch algorithm, see Table 4. Benefits are particularly evident when the problem has many parts.

Variable fixing via (14) in (KX) strengthens the continuous relaxation, thus reducing model size. On the one hand, this both improves the dual bound at root (that is, on average, 16%) and reduces the CPU time to find it (column 4 of Table 4: in most instances with 100 parts, time roughly drops by one order of magnitude); on the other hand, solver performance is enhanced: one hour CPU time takes gaps as large as 40% (on average, 15%) to less than 10% (on average 3%); moreover, gaps decrease as far as the number of parts increases.

Reducing the "big M" $f_j^2(t)$ in (GX) entails a similar benefit, and reduces on average the dual bound by 17%. This is a critical improvement: from a mean value of 33.8 (worse than the mean $z_{LB}^{\mathcal{E}}$, 38.4) the bound increases to 39.3, see column 5 in Table 2 and column 6 in Table 4. At a first glance the effect is quite surprising, as in the worst case $C^{\mathcal{E}}$ can be half the optimum C_{max}^* of the associated Bin Packing, whereas $C^{\mathcal{G}}$ takes very often advantage of the Integer Round-up Property (IRUP); however, the instances where the dual bound is improved are exactly those for which the IRUP is ineffective, and at the same time the difference between $C^{\mathcal{G}}$ and $C^{\mathcal{E}}$ is almost always close to zero. In other words, the Bin Packing polyhedron is not enough strengthened by the reformulation to compensate the looseness of the activation constraints that define L_{max}; on the other hand, $z_{LB}^{\mathcal{E}}$ implies that each part is at most fractioned into two segments.

CPU time is reduced by roughly one order of magnitude, from an average of 27 s to 2.9. This is due both to the generation of columns – that are more than halved – and to a simplification of pricing – because pricing in period t has just to do with parts due by a date $\leq t - L_{UB}$. Reducing the columns generated also improves Price-and-Branch: the average CPU time passes from 480 to 109 s, and gaps are slightly improved.

4.4. Side experiments

In the experiments of Section 4, the benchmark consists of difficult instances of 1-dimensional Bin Packing. To get further indications on the approach proposed, we tested our methods on eighty random instances generated by CUTGEN1 [13] with the following setting: $N = n \in \{20, 40, 60, 80\}$; C = stock length $= 1000$; $V1 = 0.01$, $V2 = 0.5$, that is, item lengths between 10 and 500.

As in previous tests, the combinatorial gap $z_{UB}^{S\&F} - z_{LB}^{\mathcal{E}}$ is strongly reduced (from 4.30% to 0.66%, mean values). But unlike runs on non-IRUP instances, the optimum is reached in 70 cases out of 80, which may suggest that the hardness of the underlying Bin Packing is inherited by the scheduling problem.

Please cite this article as: Arbib C, Marinelli F. Maximum lateness minimization in one-dimensional bin packing. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.06.003

On the other hand, an analysis performed to investigate the role of α_1 and α_2, suggests that the problem gets harder as far as the scheduling term of the objective function gets importance. In particular, on benchmark instances with 200 items,

- Both the combinatorial and the reformulation gap increase with α_2: the former, from 6.45% ($\alpha_2 = 0.2$) to 10.65% ($\alpha_2 = 0.8$); the latter, from 1.88% ($\alpha_2 = 0.2$) to 3.32% ($\alpha_2 = 0.8$, in this case with a reduction of CPU time).
- Also the reformulation looses tightness when α_2 grows: in the same range $[0.2, 0.8]$, the improvement $z_{LB}^{GX'}$ vs. $z_{LB}^{\mathcal{GCR}}$ decreases from 1.20% to 0.90%.
- However, the advantage of (GX′) vs. (GX) in terms of bound improvement, number of columns and CPU time, becomes more visible for larger values of α_2.

Finally, we analyzed the behavior of the Price-and-Branch algorithm in correlation with due date distribution. To this purpose, for each of the ten benchmark instances with 120 parts, we generated eleven random instances with due dates in $[0, 0.1d \cdot \lfloor C^{\mathcal{G}} \rfloor]$, $i = 0, \ldots, 10$. The optimum value is clearly non-increasing as d increases. The combinatorial approach shows gaps increasing with d; instead, Price-and-Branch appears uncorrelated with d as its gap (CPU time) variation is non-monotonic between 1.3% and 2.0% (5 and 10 s, with the exception of three peaks mainly due to the MILP solution).

5. Conclusions

We presented and exact ILP formulation and a heuristic algorithm for a 1-dimensional bin packing problem with due dates, where the objective is to minimize a convex combination of the number of bins and the maximum lateness of parts.

Cutting/packing problems with due dates are increasingly studied and have important practical applications. One quality of the approach here described is flexibility: in fact, a time-indexed formulation can easily be extended to different scheduling objectives. On the other hand, a typical drawback of this type of formulation is the quite large number of variables and constraints. The crucial role that in our experience lower and upper bounds played to strengthen activation constraints and to fix 0–1 variables suggests that, in future research, an adequate attention is to be paid to bounding effectively the objective function. Other issues, such as the design of an exact Branch-and-Price algorithm, the computation of the Pareto frontier of C_{max} vs. L_{max} and the extension of the bound strengthening technique to different scheduling objective functions deserve further investigation. Finally, future generalizations of the approach to the s-dimensional problem can take advantage from the decomposition here described.

References

[1] Adler M, Gibbons PB, Matias Y. Scheduling space-sharing for internet advertising. Journal of Scheduling 2002;5(2):103–19.
[2] Arbib C, Marinelli F. On cutting stock with due dates. Omega International Journal of Management Science 2014;46:11–20.
[3] Arbib C, Di Iorio F, Marinelli F, Rossi F. Cutting and reuse: an application from automotive component manufacturing. Operations Research 2002;50(6):923–34.
[4] Arbib C, Servilio M, Smriglio S. A competitive scheduling problem and its relevance to UMTS channel assignment. Networks 2004;44(2):132–41.
[5] Aktin T, Özdemir RG. An integrated approach to the one-dimensional cutting stock problem in coronary stent manufacturing. European Journal of Operational Research 2006;196(2):737–43.
[6] Belov G, Scheithauer G. Setup and open-stacks minimization in one-dimensional stock cutting. INFORMS Journal on Computing 2007;19(1):27–35.
[7] Bennel JA, Lee L-S, Potts CN. A genetic algorithm for two-dimensional bin packing with due dates, Int. Journal of Production Economics 2013;145(2):547–60.
[8] Carlier J, Clautiaux F, Moukrim A. New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation. Computers and Operations Research 2007;34(8):2223–50.
[9] Chen Z-L, Powell WB. Solving parallel machine scheduling problems by column generation. INFORMS Journal on Computing 1999;11(1):78–94.
[10] Clautiaux F, Alvés C, Valerio de Carvalho J. A survey of dual-feasible and superadditive functions. Annals of Operational Research 2010;179:317–42.
[11] Detti P, Agnetis A, Ciaschetti G. Polynomial algorithms for a two-class multiprocessor scheduling problem in mobile telecommunications systems. Journal of Scheduling 2005;8(3):255–73.
[12] Dresden Cutting and Packing Group (CaPaD), ⟨http://www.math.tu-dresden.de/~capad/⟩.
[13] Gau T, Wäscher G. CUTGEN1: a problem generator for the standard one-dimensional cutting stock problem. European Journal of Operational Research 1995;84(3):572–9.
[14] Giannelos NF, Georgiadis MC. Scheduling of cutting-stock processes on multiple parallel machines. Transactions of the Institution of Chemical Engineers 2001;79(Part A):747–53.
[15] Gilmore PC, Gomory RE. A linear programming approach to the cutting stock problem. Operations Research 1961;8:849–59.
[16] Hartmann S, Briskorn D. A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research 2010;1:1–14.
[17] Li S. Multi-job cutting stock problem with due-dates and release-dates. Journal of the Operational Research Society 1996;47:490–510.
[18] Marinelli F, Arbib C. Dataset of the paper maximum lateness minimization in one-dimensional bin packing. ResearchGate network; 2015. ⟨https://www.researchgate.net/profile/Fabrizio_Marinelli2⟩.
[19] Martello S, Toth P. Knapsack problems – algorithms and computer implementation. Chichester: Wiley; 1990.
[20] Pinedo ML. Scheduling: theory, algorithms, and systems. 3rd edition. Berlin: Springer-Verlag; 2008.
[21] Reinertsen H, Vossen TWM. The one-dimensional cutting stock problem with due-dates. European Journal of Operational Research 2010;201:701–11.
[22] Rosenhead J. IFORS' operational research hall of fame Leonid Vitaliyevich Kantorovich. International Transactions in Operational Research 2003;10(6):665–7.
[23] Seiden SS. On the online bin packing problem. Journal of the ACM 2002;49(5):640–71.
[24] Simchi-Levi D. New worst-case results for the bin-packing problem. Naval Research Logistics 1994;41(4):579–85.
[25] T'Kindt V, Billaut J. Multicriteria scheduling - theory, models and algorithms. 2nd edition. Berlin: Springer-Verlag; 2006.
[26] Valerio de Carvalho J. Exact solution of bin-packing problems using column generation and branch-and-bound. Annals of Operations Research 1999;86:629–59.
[27] van den Akker M, Hoogeveen H, van de Velde S. Parallel machine scheduling by column generation. Operations Research 1999;47(6):862–72.
[28] van den Akker M, Hoogeveen H, van de Velde S. Column generation. In: Desaulniers G, Desrosiers J, Solomon MM, editors. Applying column generation to machine scheduling. Springer Science; 2005. p. 303–30.
[29] van den Akker M, Hurkens CAJ, Savelsbergh MWP. Time-Indexed formulations for machine scheduling problems: column generation. INFORMS Journal on Computing 2000;12(2):111–24.
[30] Vanderbeck F. Computational study of a column generation algorithm for bin packing and cutting stock problems. Mathematical Programming 1999;86:565–94.
[31] Yazgaç T, Özdemir RG. A cutting sequencing approach to modular manufacturing. Journal of Manufacturing Technology Management 2006;15(1):20–8.

Please cite this article as: Arbib C, Marinelli F. Maximum lateness minimization in one-dimensional bin packing. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.06.003