Theoretical Computer Science 410 (2009) 3792–3803

Contents lists available at ScienceDirect

Theoretical Computer Science

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

Strip packing with precedence constraints and strip packing with release times

John Augustine [a,*,1], Sudarshan Banerjee [b,2], Sandy Irani [c]

[a] Tata Research Development and Design Centre, Pune, Maharashtra, India

[b] Center for Embedded Computer Systems, University of California at Irvine, Irvine, CA, USA

[c] Donald Bren School of Information and Computer Sciences, University of California at Irvine, Irvine, CA, USA

ARTICLE INFO

Article history:
Received 26 June 2008
Received in revised form 17 March 2009
Accepted 24 May 2009
Communicated by A. Fiat

Keywords:
Strip packing
Approximation algorithms
Precedence constraints
Linear programming
Field programmable gate array

ABSTRACT

The strip packing problem seeks to tightly pack a set of n rectangles into a strip of fixed width and arbitrary height. The rectangles model tasks and the height models time. This paper examines two variants of strip packing: when the rectangles to be placed have precedence constraints and when the rectangles have release times. Strip packing is used to model scheduling problems in which tasks require a contiguous subset of identical resources that are arranged in a linear topology. The variants studied here are motivated by scheduling tasks for dynamically reconfigurable Field-Programmable Gate Arrays (FPGAs) comprised of a linear arrangement of K homogeneous computing resources, where K is a fixed positive integer, and each task occupies a contiguous subset of these resources. For the case in which tasks have precedence constraints, we give an $O(\log n)$ approximation algorithm. We then consider the special case in which all the rectangles have uniform height, and reduce it to the resource constrained scheduling studied by Garey, Graham, Johnson and Yao, thereby extending their asymptotic results to our special case problem. We also give an absolute 3-approximation for this special case problem. For strip packing with release times, we provide an asymptotic polynomial time approximation scheme. We make the standard assumption that the rectangles have height at most 1. In addition, we also require widths to be in $[\frac{1}{K}, 1]$. For the FPGA application, this would imply that the rectangles are at least as wide as a column. Our running time is polynomial in n and $1/\epsilon$, but exponential in K.

1. Introduction

Strip packing is a relatively old and well-studied problem in the theoretical computer science literature. In this problem, a set S of rectangles must be placed within a strip of width 1 and variable height. The goal is to place the rectangles so that they do not overlap and the height of the region covered by rectangles is minimized. The cardinality of S is n and each rectangle $s \in S$ is of positive height h_s and width w_s, where $0 < w_s \leq 1$. In the classical definition of the problem and the version we study here, the rectangles can not be rotated. A *valid placement* is a specification of a point (x_s, y_s) for each $s \in S$. This point is interpreted to be the lower-left corner of the rectangle's placement in the strip. A valid placement must satisfy

* Corresponding address: Tata Research Development and Design Centre, 54B Hadapsar Industrial EstateHadapsar, 411013 Pune, Maharashtra, India. Tel.: +91 20 66086464.

E-mail addresses: jea@ics.uci.edu (J. Augustine), banerjee@ics.uci.edu (S. Banerjee), irani@ics.uci.edu (S. Irani).

1 Work done while at the University of California at Irvine.

2 Currently with Liga Systems, Sunnyvale, CA, USA.

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.05.024

the conditions that for each rectangle s, $0 \leq x_s \leq 1 - w_s$, and for all $s' \in S - \{s\}$, the rectangles s and s' do not overlap. We will use the terms *placement* and *packing* interchangeably to denote a solution to an instance of strip packing.

One application of strip packing is scheduling a resource that is arranged in a linear topology such that each task requires a contiguous subset of the resource for a fixed length of time. Each rectangle represents a task, where the width is the amount of the resource required for the task and the height is the length of time to complete the task. In this case, the width of the strip represents the total amount of the resource available normalized to 1. The goal of finding a placement of rectangles on the strip which minimizes the total height of the placement, is equivalent to minimizing the total amount of time required to complete all tasks.

The problems addressed in this paper are motivated, in particular, by modern FPGAs (field-programmable gate arrays), which consist of a rectangular grid of configurable logic blocks (along with programmable routing resources). Such devices are widely used for image processing and networking applications due to the fact that these applications have a regular structure that can be exploited on FPGAs, leading to significant performance improvement compared to an implementation on general purpose processors.

Traditionally, such devices are configured once to implement the logic functionality *before* an application starts executing. However, modern devices such as the Virtex-II device from Xilinx [25] additionally allow the device to be configured *during* application execution. This is a very powerful paradigm allowing the same set of configurable logic blocks to be reused for implementing different functions (tasks) at different points of time. This mechanism is referred to as *run-time reconfiguration* or *dynamic reconfiguration*. Due to significant engineering complexity, Virtex-II devices allow reconfiguration only along one axes — that is, any *reconfigurable* task (function) is constrained to occupy the entire height of the device. Thus, we can think of a task as occupying a contiguous set of columns of the device. Scheduling a set of tasks on such a device is equivalent to the strip packing problem — the width of the strip in which all the rectangles (tasks) are to be packed represents the width of the device. A schedule of minimum duration is simply the smallest height in which all the tasks (strips) can be packed.

Image-processing applications (such as JPEG encoding) have inherent precedence constraints between tasks. Scheduling a set of such tasks on the target computing element [4] leads to our first problem of *Strip packing with precedence constraints*. We give an approximation algorithm for this problem that guarantees a schedule whose total height is within a factor of $2 + \log(n + 1)$ of the optimal height. This result uses two simple lower bounds on the height of the optimal schedule: the sum of the areas of the rectangles to be placed and the maximum sum of the heights of rectangles along any path in the DAG representing the precedence constraints. We give an example showing that a more sophisticated lower bound will be required to achieve an approximation factor that is $o(\log n)$. We also give a 3-approximation for the special case in which all the rectangles to be placed have uniform height.

The second variant of strip packing we consider is motivated by the fact that operating systems for dynamically reconfigurable FGPAs need to consider tasks with different release times [23]. A release time for a rectangle s in a strip packing problem is a value r_s such that in any valid placement, y_s must be at least r_s. We provide an asymptotic polynomial time approximation scheme for the *strip packing with release times* under the standard assumption that the rectangles have height at most 1. There is no known APTAS, even for the strip packing problem without release times, that does not make this assumption. In addition, we also require that the widths be in $[\frac{1}{K}, 1]$, where K is the number of columns in our FPGA. This is a natural constraint in the context of FPGAs, because tasks are wide enough to span at least one column. The number of columns K is a constant and, in typical FPGAs, its value is at most 200.

1.1. Previous work

Strip packing has strong historic ties to bin packing, which has been studied extensively since the 70s. In the early 80s, Fernandez de la Vega and Lueker [8] proved the existence of an asymptotic polynomial time $(1+\epsilon)$-approximation algorithm for bin packing that was further improved by Karmarkar and Karp [15]. These works are of particular interest to us because many of our techniques are derived from them. Strip packing has also been studied extensively [3,6,2] producing several algorithms with absolute and asymptotic bounds. The best absolute bound on the approximation ratio to date is 2 and is due to Schiermeyer [22] and Steinberg [24]. The first asymptotic PTAS was due to Fernandez de la Vega and Zissimopoulos [9]. The result holds when heights are bounded from above and widths are bounded from above and below by constants. Kenyon and Rémila [16] provide an asymptotic PTAS for the general case where the widths are in $(0, 1]$. We note that our work uses several techniques from Kenyon and Remila [16], who in turn borrow from several previous works [6,8,15,9]. A lot of the original strip packing was motivated by stock cutting which did not allow rectangle rotations, because the materials often had designs that required a certain orientation. However, motivated by applications to scheduling, researchers have started studying several variants such as allowing 90-degree rotations [11], online versions of strip packing [7], exact approaches to strip packing [19] and malleability in rectangles [20]. The storage allocation problem studied by Buchsbaum et al. [5] restricts the movement of the rectangles in the strip along a single axis. Several scheduling problems have been studied under release times and precedence constraints [18,17,4,23,12]. Precedence constrained bin-packing problem variants [13,21] are of particular interest here because of their relationship to the strip packing problem with precedence constraints and uniform height. We are not aware of any theoretical work on either strip packing with release times or precedence constraints.

2. Precedence constrained strip packing

In this section, we consider the strip packing problem in which we have precedence constraints on the placement of rectangles within the strip. These precedence constraints are specified by a DAG, $G = (S, E)$ in which the vertex set is the set of rectangles. Any valid placement must satisfy the property that for each $(s, s') \in E, y_s + h_s \leq y_{s'}$. The goal, as in all strip packing problems, is to find a valid placement that minimizes $\max_{s\in S}(y_s + h_s)$. For any subset of rectangles S', we define $\mathcal{ARE A}(S')$ to be the sum of the areas of the rectangles in S': $\mathcal{ARE A}(S') = \sum_{s\in S'} h_s \cdot w_s$. We define the in-neighborhood set of a rectangle s to be all those rectangles that have an edge *into* s:

$$\mathcal{IN}(s) = \{s' \mid (s', s) \in E\}.$$

We also recursively define the function F which serves as a lower bound for the height of the top edge of a rectangle under any valid placement:

- If $\mathcal{IN}(s) = \emptyset$, then $F(s) = h_s$.
- If $\mathcal{IN}(s) \neq \emptyset, F(s) = h_s + \max_{s'\in\mathcal{IN}(s)} F(s')$.

For any $S' \subseteq S$, define $F(S') = \max_{s\in S'} F(s)$. We define $OPT(S, E)$ to be the optimal placement of rectangles that respects the precedence constraints specified by the edge set E. There are two straight-forward lower bounds on $OPT(S, E)$:

(1) $OPT(S, E) \geq F(S)$.
(2) $OPT(S, E) \geq \mathcal{ARE A}(S)$.

We use, as a subroutine, an algorithm that solves the strip packing problem without precedence constraints. Let $\mathcal{A}$ be such an algorithm. Suppose we have a set of tasks S' that have no precedence constraints between them. We will assume that a call to $\mathcal{A}(y, S')$ will assign a placement to tasks in S' according to $\mathcal{A}$ starting at a height of y in the strip. That is, for each $s \in S'$, it will assign the placement of the lower-left point of s to (x_s, y_s) such that none of the rectangles overlap and such that $\min_{s\in S'} y_s = y$. We will also assume that $\mathcal{A}(y, S')$ returns a value which is the distance from the bottom of the lowest rectangle in S' to the top of the highest rectangle in S':

$$\mathcal{A}(y, S') = \max_{s\in S'}(y_s + h_s) - y.$$

Note that since $\mathcal{A}(y, S')$ is independent of y, we will sometimes refer to it as $\mathcal{A}(S')$. We will additionally require that $\mathcal{A}$ has the property that for any set of rectangles S':

$$\mathcal{A}(y, S') \leq 2 \cdot \mathcal{ARE A}(S') + \max_{s\in S'} h_s.$$

This property is satisfied by the algorithms given in [24,22].

The algorithm with precedence constraints is defined recursively as follows. We call it $\mathcal{DC}$ since it is a Divide and Conquer strategy. The original call to $\mathcal{DC}$ would be with input $(0, S)$.

Algorithm 1 $\mathcal{DC}(y, S)$

1: If $S = \emptyset$, return 0.
2: Recalculate $F(s)$ for each $s \in S$ using the subgraph of the original DAG induced by S.
3: Assign: $H = F(S)$.
4: Assign: $S_{mid} = \{s : F(s) > H/2 \bigwedge F(s) - h_s \leq H/2\}$.
5: Assign: $S_{bot} = \{s : F(s) \leq H/2\}$
6: Assign: $S_{top} = \{s : F(s) - h_s > H/2\}$
7: Place rectangles in S_{bot} according to $\mathcal{DC}(y, S_{bot})$
8: Assign: $y = y + \mathcal{DC}(y, S_{bot})$
9: Place rectangles in S_{mid} according to $\mathcal{A}(y, S_{mid})$
10: Assign: $y = y + \mathcal{A}(y, S_{mid})$
11: Place rectangles in S_{top} according to $\mathcal{DC}(y, S_{top})$
12: Return $y + \mathcal{DC}(y, S_{top})$

The validity of the algorithm depends on the fact that all the rectangles in S_{mid} can be placed using $\mathcal{A}$ because there are no dependencies between them. This is established in the following lemma.

Lemma 2.1. *Fix an arbitrary y. Let S' be the set of rectangles s such that $F(s) > y$ and $F(s) - h_s \leq y$. Then there are no dependencies between any of the rectangles in S'.*

Proof. From the definition of $F(s)$, we can interpret it as the height of the top edge of s when placed optimally in a strip that is infinitely wide. Clearly two elements in S will be parallel to each other in any optimal placement in such a strip, thereby implying their independence. □

We need one more lemma before we are ready to prove the bound on $\mathcal{DC}$:

Lemma 2.2. *The set S_{mid} can not be empty.*

Proof. From the recursive nature of the definition of $F(s)$, it is clear that there is a "tight" dependency path from some rectangle $s_{base}|\mathcal{IN}s_{base} = \emptyset$ to another rectangle $s_{top}|F(s_{top}) = H$ such that sum of the heights of the rectangles in this path is H. Clearly, one of the rectangles in this path must be in S_{mid}. □

Now we can establish the bound on the height of the schedule produced by $\mathcal{DC}$.

Theorem 2.3. $\mathcal{DC}(S) \leq (2 + \log(n+1))OPT(S, E)$.

Proof. We will show that $\mathcal{DC}(S) \leq \log(n+1) \cdot F(S) + 2 \cdot \mathcal{ARE}\mathcal{A}(S)$. We prove the result by induction on the number of tasks in S. Clearly if S has only one task or is empty, the statement above holds.

Let $n_{bot} = |S_{bot}|$ and $n_{top} = |S_{top}|$. By Lemma 2.2, we know that $n_{top} + n_{bot} < n$. By induction, we know that

$$\mathcal{DC}(S_{bot}) \leq \log(n_{bot} + 1) \cdot F(S)/2 + 2 \cdot \mathcal{ARE}\mathcal{A}(S_{bot})$$

and

$$\mathcal{DC}(S_{top}) \leq \log(n_{top} + 1) \cdot F(S)/2 + 2 \cdot \mathcal{ARE}\mathcal{A}(S_{top}).$$

Furthermore, by the bound on $\mathcal{A}$, we know that $\mathcal{A}(S_{mid}) \leq 2\mathcal{ARE}\mathcal{A}(S_{mid}) + F(S)$. We have that

$$\begin{aligned}
\mathcal{DC}(S) &= \mathcal{DC}(S_{bot}) + \mathcal{A}(S_{mid}) + \mathcal{DC}(S_{top}) \\
&\leq 2(\mathcal{ARE}\mathcal{A}(S_{bot}) + \mathcal{ARE}\mathcal{A}(S_{mid}) + \mathcal{ARE}\mathcal{A}(S_{top})) + F(S) + (\log(n_{bot} + 1) + \log(n_{top} + 1))F(S)/2 \\
&\leq 2\mathcal{ARE}\mathcal{A}(S) + (2 + \log(n_{bot} + 1) + \log(n_{top} + 1))F(S)/2 \\
&\leq 2\mathcal{ARE}\mathcal{A}(S) + \log(n+1) \cdot F(S).
\end{aligned}$$

The last inequality holds because for any non-negative integers n_1, n_2 and n such that $n_1 + n_2 < n$ and $n \geq 2$, $\log(n_1 + 1) + \log(n_2 + 1) \leq 2(\log(n+1) - 1)$. □

2.1. A bottleneck to an $o(\log n)$-approximation

The following example illustrates the existence of a class of problem instances for which the optimal placement is an $\Omega(\log n)$ factor larger than the two simple lower bounds ($\mathcal{ARE}\mathcal{A}(S)$ and $F(S)$) that we use here. This demonstrates that a more sophisticated lower bound for the optimal placement will be required in order to achieve an approximation factor that is $o(\log n)$.

Lemma 2.4. *There exist arbitrary sized instances of the strip packing problem with precedence constraints such that*

$$OPT(S, E) \in \Omega(\log n) \max_{s \in S} F(s)$$

and

$$OPT(S, E) \in \Omega(\log n)\mathcal{ARE}\mathcal{A}(S).$$

Proof. Without loss of generality, we assume $n = 2^{k+1} - 2$ for some positive integer k. Given n, we construct the instance comprising of a set of rectangles S that can be partitioned into two subsets S_{tall} and S_{wide} that we also call tall and wide rectangles respectively. The two sets are of equal cardinality, although some of the wide rectangles are unused in constructing the example. The basic construction is illustrated in Fig. 1. Each rectangle $s \in S_{wide}$ (shown as unshaded rectangles in Fig. 1), has $h_s = \epsilon \to 0$ and $w_s = 1$. The tall rectangles are the shaded rectangles in Fig. 1. S_{tall} is sorted in non-increasing order of heights and the height of the ith rectangle in S_{tall} is $1/(2^{\lfloor \log i \rfloor})$, $1 \leq i \leq n/2$. Notice that there are k distinct heights and there are 2^{i-1} rectangles of height $1/2^{i-1}$ for $1 \leq i \leq k$. All rectangles in S_{tall} have a width of $1/k$. The precedence constraints are as illustrated in Fig. 1. There are k chains in $G(S, E)$ and each chain i is comprised of all the rectangles of a given height $h_s = 1/2^{i-1}$ in an arbitrary order. In addition, each contiguous pair of elements in S^i_{tall} sandwich a rectangle from S_{wide}. It is easy to see that the number of rectangles needed in S_{wide} is at most $n/2$ and the extra rectangles form a separate chain that will not cause any significant increase in $OPT(S, E)$. Notice that $\max_{s \in S} F(s) = \mathcal{ARE}\mathcal{A}(S) = 1$, as $\epsilon \to 0$. However, the wide rectangles hinder us from packing them densely because they force us to pack in shelves and obstruct any tall rectangle from spanning multiple shelves due to the fact that their width is 1. The task in chain 1 in the $G(S, E)$ dominates one of the shelves which is of height 1. Since both rectangles in chain 2 cannot be placed in parallel with chain 1, we need to create at least 1 more shelf of height $1/2$. Every time we place a new chain i, we can place at most half of the rectangles in shelves that we have already created. Recall that chain i has 2^{i-1} rectangles of height $1/2^{i-1}$ and hence we create 2^{i-2} shelves increasing the height by $1/2$. Since there are k chains, the total height is at least $k/2 \in \Omega(\log n)$. □

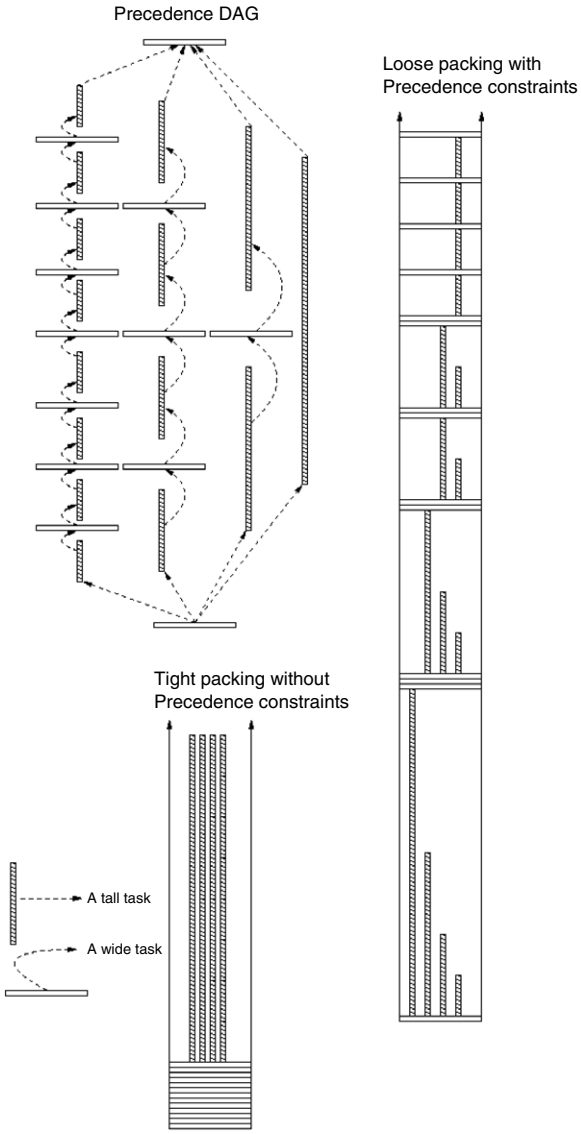

Fig. 1. Illustrates the difficulty in proving that an algorithm is $o(\log n)$-approximate.

2.2. Fixed height

In this section, we study precedence constrained strip packing when the heights of the rectangles are fixed and therefore can be normalized to 1. We first reduce it to the precedence constrained bin-packing problem studied as a special case of the resource constrained scheduling problem by Garey, Graham, Johnson and Yao [13] and show that their asymptotic results carry over to our problem. In the precedence constrained bin-packing problem, we have n tasks each with a size in $(0, 1]$ associated with it. We also have a sequence of bins and each bin can hold tasks that have a total size of at most 1. Each task should be placed in a bin such that the number of bins needed is minimized. In addition, we are given a partial order $\prec$ on the tasks such that if $a \prec b$, then we need to place a in a bin that is strictly earlier in the sequence than b. Garey et al. provide an asymptotic 2.7-approximation, i.e. the cost of their solution is at most 2.7 times the cost of the optimal solution plus a constant. We show that this result applies to strip packing with rectangles of fixed height. We then provide an absolute 3-approximation algorithm (with no additive constant). We are unaware of any other result for the precedence constrained bin-packing problem that might improve on our result in the absolute sense.

Consider precedence constrained strip packing with rectangles of height 1. Define a shelf i for some positive integer i to be the portion of the strip from height $(i - 1)$ to i. We say that a solution to the strip packing problem is a *shelf* solution if each rectangle is placed within a single shelf. We will show that we can restrict our attention to shelf solutions by showing that any solution for the strip packing problem can be converted to a shelf solution without increasing the total height of the placement. Recall that x_s refers to the height of the lower left corner of rectangle s. The conversion is achieved by iteratively

picking the rectangle s that spans two shelves and has the lowest x_s, and sliding it down until it fits in the lower of the two shelves that it spans. The only way we cannot slide it down is if there was another rectangle obstructing it, but such a rectangle s' would also be spanning two shelves and $x_{s'} < x_s$, which will be a contradiction. We repeat this process of sliding down until all rectangles are contained inside a shelf. Thus we have established that we can restrict ourselves to shelf algorithms, i.e., algorithms that arrange the rectangles in shelves. It is quite easy to see that if we consider the shelves to be bins and each rectangle of width w to be a task of size w, the two problems are equivalent. Therefore, the asymptotic 2.7-approximation for precedence constraint bin packing applies to precedence constrained strip packing with rectangles of uniform height.

Now we describe an algorithm $\mathcal{F}$, a natural variant of the Next Fit algorithm for bin packing, that achieves an absolute 3-approximation. The algorithm $\mathcal{F}$ constructs a shelf placement. It keeps one open shelf at the top of the current placement. All shelves below the open shelf are closed. A rectangle is said to be *available* if all of its predecessors in $G(S, E)$ have already been placed on closed shelves. Algorithm $\mathcal{F}$ maintains a queue of all available rectangles and will keep taking rectangles from the head of the queue, placing them from left to right on the open shelf. This continues until the rectangle at the head of the queue can not fit on the open shelf or the queue is empty. When this happens, the algorithm closes the currently open shelf and opens a new shelf above it. Every time a new shelf is opened, the ready queue becomes repopulated (as new rectangles may have become available) and the process of fitting rectangles on the currently open shelf begins again. We repeat this process until we exhaust all of the rectangles. If the current open shelf is closed because the ready queue is empty, we call this a *skip*. The following lemma bounds the number of skips.

Lemma 2.5. *The number of skips performed by $\mathcal{F}$ is at most $OPT(S, E)$.*

Proof. First observe that if there is path of length p in $G = (S, E)$, then p is a lower bound for $OPT(S, E)$. Define a *skip-shelf* to be a shelf which resulted in a skip. In other words, the ready list is empty after placing tasks in a skip-shelf. Let L be the set of rectangles that are placed on any skip-shelf. For any rectangle s that is placed above L, there must be a path in the DAG from some rectangle in L to s. If not, s would be in the ready list just after all the rectangles in L had been placed on the shelf. Whereas the fact that the shelf resulted in a skip means that the ready list was empty.

We will construct a path in the DAG with a vertex corresponding to a rectangle in each skip-shelf. Start by selecting an arbitrary rectangle s in the top-most skip-shelf. This will be the current rectangle. There must be a rectangle in the next highest skip-shelf that has a path to s. Call this rectangle the current rectangle and continue downwards until the bottom skip-shelf is reached. By concatenating these paths, we construct a path with a vertex in each skip-shelf. □

Theorem 2.6. *$\mathcal{F}$ has an approximation ratio of* 3 *for strip packing with precedence constraints when rectangles have uniform height.*

Proof. We employ a method of coloring the shelves by sweeping from bottom to top in the strip. To start with, the current shelf is the first shelf. If current is shelf i and the total area covered by rectangles in shelves i and $i+1$ is greater than or equal to 1, then color shelves i and $i+1$ red and move the current to $i+2$. Otherwise, color the shelf green and move on to shelf $i+1$. Notice that the red shelves have a density of coverage at least $1/2$. Also notice that each green shelf is a skip-shelf. Suppose that we have a green shelf which did not cause a skip step, then the queue was not empty at the time the shelf was closed and the rectangle at the head of the queue must have not fit onto the shelf. However, since this rectangle was the first to be placed onto shelf $i+1$, this would mean that the total area on shelves i and $i+1$ must exceed 1 and the shelf would have been colored red.

Let r and g be the number of red and green shelves respectively. We know that the total height of the packing produced by $\mathcal{F}$ is $(r+g)$. Since red shelves have density at least $1/2$, $r \leq 2\mathcal{ARE}\mathcal{A}(S) \leq 2 \cdot OPT(S, E)$. We also have that $g \leq OPT(S, E)$ by Lemma 2.5. Therefore, $(r+g) \leq 3 \cdot OPT(S, E)$. □

Similar to Lemma 2.4, Lemma 2.7 complements our algorithm with approximation ratio 3 by providing evidence for an inherent difficulty in proving an algorithm to be ρ-approximate for some $\rho < 3$. Recall that $\mathcal{ARE}\mathcal{A}(S)$ and $F(S)$ are the two straightforward lower bound on the optimal height of a packing for precedence constrained strip packing regardless of restrictions on rectangle heights.

Lemma 2.7. *For all $n = 3k$, where k is a positive integer, and for all arbitrarily small ϵ, there exist instances of the strip packing problem with precedence constraints and uniform height such that the number of rectangles is n and*

$$OPT(S, E) = 3(\max_{s \in S} F(s) - 1)$$

and

$$OPT(S, E) = 3 \cdot \mathcal{ARE}\mathcal{A}(S) - 3n\epsilon.$$

Proof. We have two types of rectangles in our construction as illustrated in Fig. 2. A third (of the total n rectangles) are narrow rectangles of height 1 and width $\epsilon \to 0$ and the rest are wide rectangles, the shaded rectangles in Fig. 2, of height 1 and width $1/2 + \epsilon$. The precedence constraints are as shown in the figure. The narrow rectangles form a single chain, which requires that they be placed one on top of the other. Each wide rectangle has an edge to the first narrow rectangle in the

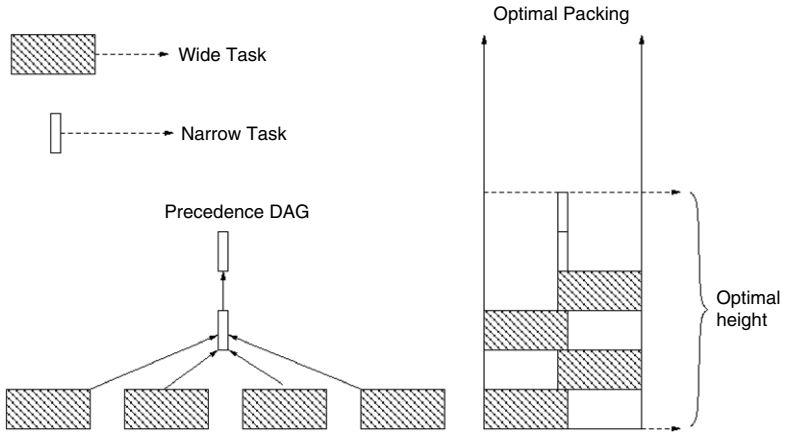

Fig. 2. Illustrates Lemma 2.7.

path, requiring that all the wide rectangles be placed below all the narrow rectangles. The total area $\mathcal{ARE}\mathcal{A}(S)$ is the sum of the areas of narrow and wide rectangles and it is given by

$$\begin{aligned}\mathcal{ARE}\mathcal{A}(S) &= \frac{2n}{3}(1/2+\epsilon)+\frac{n\epsilon}{3}\\ &= \frac{n}{3}+n\epsilon.\end{aligned}$$

The total height of the critical path is given by

$$\max_{s\in S} F(s) = \frac{n}{3}+1.$$

The optimal packing, as shown in the figure, places the rectangles in series, thereby forcing the optimal height to n. This proves our lemma. □

3. Strip packing with release times

In this section, we constrain our problem by associating a release time r_s with each $s \in S$. The base of the strip is height 0 and each rectangle s has to be placed at or above height r_s in the strip. A valid packing is defined as a placement of each rectangle in S in the strip such that the rectangles are contained entirely in the strip and there is no overlapping of rectangles. The rectangles cannot be broken into smaller pieces in any fashion; from the perspective of placing jobs on FPGAs, we disallow preemption or placing any job in non-contiguous columns. As usual, we are trying to minimize the maximum height given by $\max_{s\in S}(y_s + h_s)$, where y_s is the position of the base of the rectangle s in the strip packing and $h_s \leq 1$ by assumption. We also assume that the widths of the rectangles are in the interval $[\frac{1}{K}, 1]$ for some constant K. The running time of the algorithm will be exponential in K. When the rectangles model jobs on an FPGA with K columns, this assumption enforces each job to be at least one column wide — a natural requirement.

Let P denote the input problem instance. P is a set of rectangles, each with a specified width, height and release time. We provide an asymptotic PTAS for P. We seek to use linear programming (LP) to solve P, but since it is integral in nature, we cannot directly apply LP. Therefore, we convert it to a fractional version in which each rectangle can be sliced horizontally and the resulting pieces can be placed in separate locations. Additionally, the fractional version permits the pieces resulting from a particular rectangle to be placed in parallel. That is, there can be more than one piece of the same rectangle that overlaps the same horizontal line across the strip. However, all the pieces of a rectangle must be placed at or above the release time of the rectangle. For any input P to the original problem, let $OPT(P)$ denote the optimal solution for the original integral version of the problem and $OPT_f(P)$ denote the optimal solution for the fractional version of the problem.

We will also make use of two reductions. First, we reduce P, the original input, to $P(R)$, such that the rectangles in $P(R)$ have only R distinct release times. Then we reduce $P(R)$ to $P(R, W)$ such that in $P(R, W)$, the rectangles have one of R distinct release times and one of W distinct widths. In each case, the cost of the optimal fractional solution increases only by a small amount. In addition, there is a one-to-one correspondence between rectangles in the original problem and the constrained problem. Furthermore, the widths and release times of each rectangle will stay the same or increase in the course of the reduction. Thus, when we finally convert the fractional solution of $P(R, W)$ to an integral solution, this will yield a solution to the original problem instance P. This process is outlined in Algorithm 2. The output in line 9 of Algorithm 2 is a packing for $P(R, W)$, but this can be easily adapted to P because, by construction, the release times are higher and the widths are wider for each input rectangle in $P(R, W)$ than in P.

Algorithm 2 APTAS for strip packing with release times

1: **INPUT:** An instance P and error parameter ϵ
2: Assign: $\epsilon' := \epsilon/3$
3: Assign: $R := \lceil 1/\epsilon' \rceil$
4: Assign: $W := \lceil 1/\epsilon' \rceil K(R+1)$
5: **Reduce** P to $P(R)$ according to the proof of Lemma 3.1
6: **Reduce** $P(R)$ to $P(R, W)$ according to the proof of Lemma 3.2
7: **Solve** Use linear programming to get $OPT_f(P(R, W))$ as outlined in the proof of Lemma 3.3
8: **Convert** fractional solution $OPT_f(P(R, W))$ to integral solution $\mathcal{S}(R, W)$ according to the proof of Lemma 3.4
9: **OUTPUT:** $\mathcal{S}(R, W)$

3.1. Constructing $P(R, W)$ from P

We provide a series of lemmas that describe key insights to show how $OPT_f(P(R, W))$ can be modified to work as an asymptotic PTAS for P. Lemmas 3.1 and 3.2 are proven constructively, thus providing constructive details for lines 5 and 6 respectively. Lemma 3.1 bounds the number of release times.

Lemma 3.1. *For a fixed $\epsilon_r > 0$, problem instance P can be reduced to an instance $P(R)$, where $R = \lceil 1/\epsilon_r \rceil$ such that $OPT_f(P(R)) \leq (1+\epsilon_r)OPT_f(P)$. This reduction takes time polynomial in the size of the input and $1/\epsilon_r$. Furthermore, the set of rectangles in P and $P(R)$ is the same and the release time of each rectangle in $P(R)$ is no earlier than its release time in P.*

Proof. We provide a constructive proof that bounds the number of release times similar to [1]. Define $r_{\max} = \max_{s \in S} r_s$. Note that $r_{\max}$ is a lower bound for any solution of $\mathcal{S}$. Define $\delta = \epsilon_r r_{\max}$. We define $\varrho_j = j \cdot \delta$, for $0 \leq j \leq 1/\epsilon_r$. We now define two problem instances based on P:

$P^{\downarrow}$: For each rectangle $s \in S$ defined in the problem instance P, we have a corresponding rectangle $s^{\downarrow}$ in $P^{\downarrow}$ with the same dimensions as in s, but its release time is $\max_{0 \leq i < 1/\epsilon_r} \varrho_i \leq r_s$. Also, define the optimal fractional packing for this problem to be $OPT^{\downarrow}$.

$P^{\uparrow}$: For each rectangle $s^{\downarrow}$ in $P^{\downarrow}$ defined in the problem instance $P^{\downarrow}$, we have a corresponding rectangle $s^{\uparrow}$ in $P^{\uparrow}$ with the same dimensions as in $s^{\downarrow}$, but its release time is $r_{s^{\downarrow}} + \delta$. Also, define the optimal fractional packing for this problem to be $OPT^{\uparrow}$.

From the definitions, we can see that any valid solution to $P^{\downarrow}$ can be pushed up by δ to form a valid solution for $P^{\uparrow}$ and vice versa. Therefore,

$$OPT^{\uparrow} \leq OPT^{\downarrow} + \delta. \tag{3.1}$$

In addition, each rectangle s in P has release times sandwiched in $[r_{s^{\downarrow}}, r_{s^{\uparrow}}]$. It follows that $OPT^{\downarrow} \leq OPT \leq OPT^{\uparrow}$. Hence,

$$OPT^{\uparrow} \leq OPT + \delta = OPT + \epsilon_r r_{\max} \leq (1+\epsilon_r)OPT.$$

$P^{\uparrow}$, which obviously can be constructed in time polynomial in the input size and $1/\epsilon$, serves as $P(R)$, the reduced problem instance that has exactly R distinct release times. □

The following lemma enables us to reduce our problem with a bounded number of release times to one with a bounded number of release times and rectangle widths while only suffering a small increase in the total height of the placement.

Lemma 3.2. *Given constant positive integers R and W such that W is an integer multiple of $R+1$, a problem instance $P(R)$ can be reduced in time polynomial in the size of the input to an instance $P(R, W)$ such that*

$$OPT_f(P(R, W)) \leq OPT_f(P(R)) \cdot \left(1 + \frac{(R+1)K}{W}\right).$$

Proof. This constructive proof uses the grouping technique previously employed in [8,9,16], but we modify it to account for the R release times. It will be apparent that the construction takes time polynomial in the size of the input, R, and W. Recall that the release times are denoted by $\varrho_0 = 0 < \varrho_1 < \varrho_2 < \cdots < \varrho_R < \varrho_{R+1} = \infty$. We first outline the reduction and then show that the lemma holds for it. Essentially, for the jobs in each release time, we construct an instance whose jobs have W/(R+1) distinct widths in the same manner as in [16]. We then ensure that this construction at each release time can be extended to the case with multiple release times. For the sake of completeness, we provide all the details of the construction.

Consider the stacking of a set of rectangles S such that the rectangles are placed left justified one on top of another sorted in non-increasing order of their widths from bottom to top. An illustrative example is shown in Fig. 3. We call this the *stacking of S*. Consider two sets S and S' of rectangles all of which have the same release time. We say that the stacking of S *is contained in* the stacking of S' if the area covered by the rectangles in S' when they are stacked can be placed so that it completely covers the area of the stacking of S. Finally a set of rectangles P is *contained* in another set of rectangles P' if for

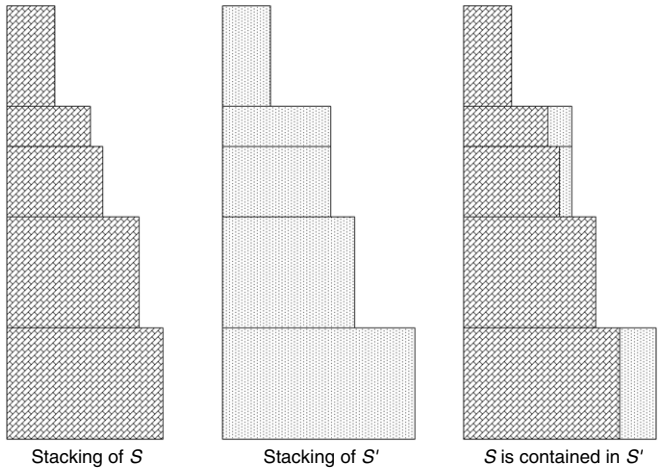

Fig. 3. Figure illustrating the definition of a stacking of S. Assume that S and S' have the same release times. Note that S is contained in S'.

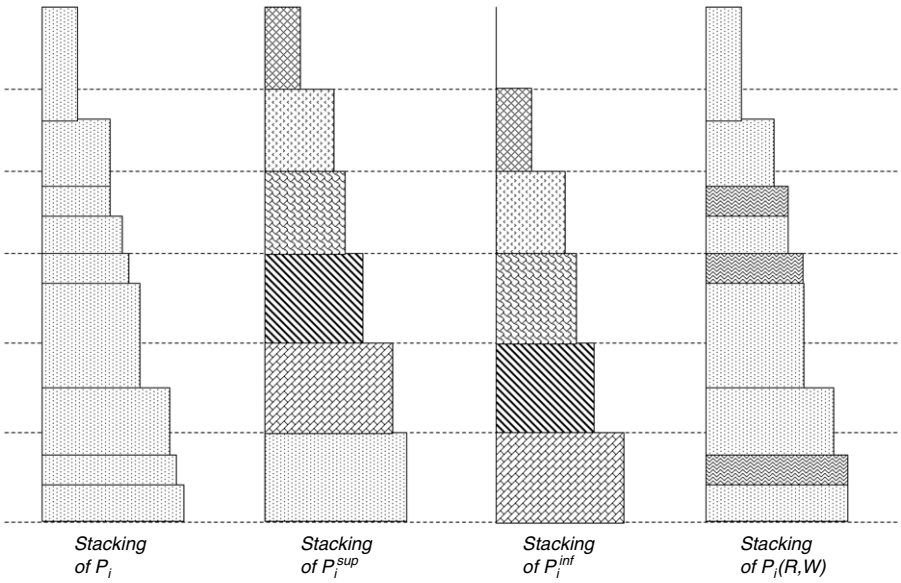

Fig. 4. Figure illustrating P_i, P_i^{sup}, P_i^{inf}, and $P_i(R, W)$. The dotted horizontal lines are of the form $y = \ell H(P_i)(R+1)/W$ for $0 \leq \ell < W/(R+1)$. The densely patterned rectangles in $P_i(R, W)$ are wider than their counterparts in P_i. In P_i^{sup} and P_i^{inf}, the rectangles of equal widths have the same pattern.

each distinct release time ϱ, the stacking of the set of jobs with release time ϱ in P is contained in the stacking of the set of jobs with release time ϱ in P'. Note that if P is contained in P', then $OPT_f(P) \leq OPT_f(P')$.

We construct an instance $P(R, W)$ (a set of rectangles with associated release time, height and width) such that the rectangles in $P(R, W)$ have at most W distinct widths and $P(R)$ is contained in $P(R, W)$. We partition $P(R) = P_0 \bigcup P_1 \bigcup \cdots \bigcup P_R$, where each $P_i = \{s : s \in P(R) \bigwedge r_s = \varrho_i\}$, $0 \leq i \leq R$. Let $H(P_i)$ denote the height of stacking of the rectangles in P_i. We use $W/(R+1)$ out of the W widths per part P_i of $P(R)$. Hence, we divide each stacking of the parts $P_i \subseteq P(R)$ into $W/(R+1)$ equal pieces by cutting the stack with lines $y = \ell H(P_i)(R+1)/W$, where $0 \leq \ell < W/(R+1)$. A rectangle is a threshold rectangle if a line $\ell H(P_i)(R+1)/W$ cuts through its interior or aligns with its base. The threshold rectangles partition the stacking of P_i into groups, i.e., each group starts with a threshold rectangle in the stack and includes all rectangles above it until but excluding the next threshold rectangle above it (if it exists). Let $P_{i,\ell}$ denote the set of rectangles in group ℓ of P_i. Let $w_{i,\ell}$ denote the width of the threshold rectangle in that group. Note that width of all the rectangles in $P_{i,\ell}$ is at most $w_{i,\ell}$. For ease of notation, we will define $w_{i,W/(R+1)} = 0$. Similar to P_i, we define $P_i(R, W) = \{s : s \in P(R, W) \bigwedge r_s = \varrho_i\}$, $0 \leq i \leq R$. To construct each $P_i(R, W)$, we set the widths of all the rectangles in $P_{i,\ell}$ to be $w_{i,\ell}$ for each $0 \leq i < R$ and $0 \leq \ell < W/(R+1)$. The rest of the attributes of each rectangle such as its height and release time remain unaltered. We can easily combine the parts, $P_i(R, W)$, to get $P(R, W)$. We now need to bound $OPT_f(P(R, W))$ to prove the lemma (refer Fig. 4).

We consider two other sets of rectangles $P^{inf} = \bigcup_i P_i^{inf}$ and $P^{sup} = \bigcup_i P_i^{sup}$. Each P_i^{inf} will consist of $W/(R+1)$ rectangles each with height $H(P_i)(R+1)/W$, release time ϱ_i and width $w_{i,\ell+1}$ where $0 \leq \ell < W/(R+1)$. Similarly, P_i^{sup} will consist of $W/(R+1)$ rectangles each with height $H(P_i)(R+1)/W$, release time ϱ_i and width $w_{i,\ell}$. The four instances are related in that P^{inf} is contained in $P(R)$ (the original set of input rectangles) which is contained in $P(R, W)$ which is in turn contained in P^{sup}. Thus, we have that

$$OPT_f(P^{inf}) \leq OPT_f(P(R)) \leq OPT_f(P(R, W)) \leq OPT_f(P^{sup}).$$

We need only to show now that

$$OPT_f(P^{sup}) \leq OPT_f(P(R)) \cdot \left(1 + \frac{(R+1)K}{W}\right).$$

The sets P^{inf} and P^{sup} are almost the same except that P^{sup} has R extra rectangles, each of width at most 1 and height $H(P_i)(R+1)/W$, corresponding to each release time ϱ_i. The optimal fractional strip packing schedules for the two differ by at most $\sum_i H(P_i)(R+1)/W = H(P(R))(R+1)/W$. Therefore,

$$OPT_f(P^{sup}) \leq OPT_f(P^{inf}) + H(P(R))(R+1)/W.$$

Since the widths of the rectangles are lower bounded by $\frac{1}{K}$, $H(P(R)) \cdot \frac{1}{K} \leq \mathcal{ARE}\mathcal{A}(P(R)) \leq OPT_f(P(R))$. Therefore, we have

$$\begin{aligned} OPT_f(P^{sup}) &\leq OPT_f(P^{inf}) + H(P(R))(R+1)/W \\ &\leq OPT_f(P(R)) + K \cdot OPT_f(P(R))(R+1)/W \\ &\leq OPT_f(P(R))\,(1 + K(R+1)/W)\,. \quad \square \end{aligned}$$

3.2. Constructing an Integral Solution from $OPT_f(P(R, W))$

We can formulate an instance of the fractional problem with a constant number of widths and release times as a linear programming problem. The widths and release times are denoted by ω_i, $1 \leq i \leq W$ and ϱ_j, $1 \leq j \leq R$, respectively. The release times must be in ascending order. In order to use ϱ_R as a lower bound on the packing, we assume that there is at least one rectangle that has a release time of ϱ_R. For ease of notation, we define an extra $\varrho_{R+1} = \infty$. Define phase j to be the time between ϱ_j to ϱ_{j+1}.

We now present the concept of *configurations* that was initially employed in the context of bin-packing [8] and subsequently used in strip packing as well [8,9,16]. We define a configuration as a multiset of widths such that the widths sum to a value less than 1. Let C be the set, with cardinality Q (which is exponential in $\frac{1}{K}$), of all possible configurations that can be formed using the allowed widths. Intuitively, each configuration is a possible combination of widths that can be contained within the strip at any fixed height. In a valid optimal packing, each configuration has a non-negative height associated with it, although the entire height of a particular configuration may not be in one contiguous piece, but rather spread across several phases. We show that the above fractional packing problem can be defined by a linear programming formulation. The linear program will determine the height allocated to each configuration q during each phase j. For this, we define $Q(R+1)$ variables x_j^q, where $1 \leq q \leq Q$ and $0 \leq j \leq R$. The variable x_j^q refers to the height assigned to configuration q in phase j. It will be convenient to refer to vector X_j as the variables pertaining to phase j: $X_j = (x_j^1, x_j^2, \ldots, x_j^Q)$. Our objective will be to minimize any height that is assigned to the packing in excess of the final release time, i.e., in phase R. Hence, our objective function is

$$\min \sum_q x_R^q \tag{3.2}$$

subject to constraints that ensure validity of the assignment. Trivially, we know that each $x_j^q \geq 0$. In addition, the sum of the heights associated with each phase j cannot exceed $\varrho_{j+1} - \varrho_j$, i.e., we cannot pack more than is allowed in each phase. Therefore, we can write the *packing constraints* as follows for every integer $j \in [0, R]$:

$$\sum_q x_j^q \leq \varrho_{j+1} - \varrho_j. \tag{3.3}$$

Finally, we have to ensure that the input rectangles are completely covered, i.e. the sum of all the heights of rectangles of width say ω_i released after some release time ϱ_j should be accounted for in the linear programming constraints. In order to facilitate this, we define matrix A of size $W \times Q$ with elements a_{iq} being the number of occurrences of width ω_i in configuration q. Thus $\sum_q a_{iq} \cdot x_j^q$ (the ith entry in $A \cdot X_j$) is equal to the total height of tasks of width ω_i that get packed in phase j. We need to ensure that the total height of tasks of width ω_i that gets packed into phases k through R is at least the sum of the heights of the rectangles of width ω_i whose release time happens on or after r_k. We define vector $B_j = (b_j^1, b_j^2, \ldots, b_j^W)$ such that each b_j^i equals the sum of heights of all the rectangles that are released at ϱ_j and has width ω_i. More formally,

$$b_j^i = \sum_{\{s:(w_s=\omega_i) \bigwedge (r_s=\varrho_j)\}} h_s.$$

We need that for each $0 \leq k \leq R$:

$$\sum_{j=k}^{R} A \cdot X_j \geq \sum_{j=k}^{R} B_j. \tag{3.4}$$

Note that the expressions in the inequality above are vectors, each with R entries, and the inequality must hold for each individual entry. We denote the above linear programming formulation consisting of the objective function in (3.2) and constraints in (3.3) and (3.4) along with the non-negativity constraint on the variables as $\mathcal{LP}$. Note that the input size of an instance of $\mathcal{LP}$ is polynomial in the number of input rectangles, R, and W, but exponential in $\frac{1}{K}$.

For any solution to the $\mathcal{LP}$ defined above, we say that configuration q *occurs* in phase j if $x_j^q > 0$. Two *occurrences* are distinct if they are different configurations or if they occur in different phases.

Lemma 3.3. *The sum of ϱ_R and the optimal solution to the linear programming formulation $\mathcal{LP}$ will be the height of the optimal packing $OPT_f(P(R, W))$. In addition, any feasible solution to $\mathcal{LP}$ can be used to construct the solution to the fractional packing problem. Furthermore, the optimal solution to $OPT_f(R, W)$ uses only $(W + 1)(R + 1)$ distinct occurrences of configurations.*

Proof. A similar flavor of linear programming formulation without the packing constraints has been employed in the past by several researchers working on related problems [8,15,16]. Given an optimal solution to $\mathcal{LP}$, we simply construct a fractional solution to the strip-packing problem on instance $P(R, W)$, by placing each configuration in the appropriate phase and allocating the height that the solution produced as the x value for that configuration. The covering constraints ensure that all the widths are adequately represented in a fractional manner and the packing constraints ensure that the sum of the heights of configurations allocated per phase do not exceed the height of that phase. This will be optimal because the $\mathcal{LP}$ seeks to minimize the height in excess of the last release time ϱ_R.

Finally, since the optimal solution is a basic solution, and our $\mathcal{LP}$ has $(W + 1)(R + 1)$ constraints, we will have at most $(W + 1)(R + 1)$ non-zero x_i^q values. Therefore, the optimal fractional solution to $P(R, W)$ uses only $(W + 1)(R + 1)$ distinct configurations. Note that techniques in [10,14] can be used to solve $\mathcal{LP}$, and since they provide an optimal solution, the number of non-zero variables that they produce will be at most $(W + 1)(R + 1)$. □

Lemma 3.4 allows us to convert the fractional solution to an integral one with an additive increase to the total height of the packing that is bounded by the number of distinct occurrences of configurations in the solution. We note here that the technique used to prove Lemma 3.4 is well-established in bin packing and strip packing literature [8,16]. We provide the full proof for the sake of completeness. Since the number of configurations used is bounded by $(W + 1)(R + 1)$ (by Lemma 3.3), this will result in an additive increase of $(W + 1)(R + 1)$ in the height of the final integral solution we obtain.

Lemma 3.4. *If a solution $OPT_f(\mathcal{S})$ for an instance $\mathcal{S}$ uses at most k occurrences of configurations, then it can be converted to an integral solution $OPT(\mathcal{S})$ such that $OPT(\mathcal{S}) \leq OPT_f(\mathcal{S}) + k$.*

Proof. For each $x_j^q > 0$, we reserve an area of width 1 and height x_j^q for configuration q in between r_j and r_{j+1}. In course of placing rectangles in the integral solution, the reserved areas may shift upwards.

To convert the fractional solution to an integral solution, consider each reserved area from bottom up. Suppose that we are currently working on the reserved area associated with variable x_j^q. Each occurrence of width ω_i in the configuration q makes a column of width ω_i and height x_j^q. Greedily fill the column with available rectangles of width ω_i until the column is completely filled or until there are no rectangles of width ω_i. The last rectangle placed may extend beyond the column but not by more than 1. Repeat this process for each occurrence of each width in the configuration. The placement of rectangles within the columns associated with this reserved area will take up a height of at most $1 + x_j^q$. Expand the reserved area for this occurrence so that it covers all the rectangles which have been assigned to it. Shift the reserved areas above it upwards to accommodate the expansion. Note that a reserved area is shifted up by at most r if there are r reserved areas below it. Continue this process until all the rectangles are assigned. The top of the last rectangle placed in this manner will have height at most

$$\varrho_R + \sum_q x_R^q + k \leq OPT_f(\mathcal{S}) + k,$$

thereby proving the lemma. □

Theorem 3.5. *Algorithm 2 is an asymptotically $(1 + \epsilon)$-approximate algorithm for P and runs in time polynomial in n, $1/\epsilon$.*

Proof. Assign $\epsilon' = \epsilon/3$, $R = \lceil 1/\epsilon' \rceil$, and $W = \lceil 1/\epsilon' \rceil (R + 1)K$.

It is obvious that the running time is polynomial in n and $1/\epsilon$. By Lemma 3.3, the solution to the fractional problem obtained in Step 7 uses at most $(W + 1)(R + 1)$ configurations. By Lemma 3.2 we know that

$$OPT_f(P(R, W)) \leq \left(1 + \frac{K \cdot (R + 1)}{W}\right) \cdot OPT_f(P(R)) \leq (1 + \epsilon')OPT_f(P(R)).$$

By Lemma 3.1, we have that

$$OPT_f(P(R)) \leq (1 + \epsilon') \cdot OPT_f(P).$$

Therefore,

$$OPT_f(P(R, W)) \leq (1 + \epsilon')^2 \cdot OPT_f(P) \leq (1 + \epsilon)OPT_f(P).$$

Finally, we convert the fractional solution to an integral solution. The fractional solution we have has at most $(W + 1)(R + 1)$ configurations. Therefore, from Lemma 3.4, we get

$$S(R, W) \leq (1 + \epsilon)OPT(f) + (W + 1)(R + 1).$$

Since W and R are dependent only on ϵ and K, our theorem is proved. □

Acknowledgments

The authors are grateful to George Lueker for providing us with some key insights. We thank Elisabeth Guenther for her comments on an earlier manuscript. We are also thankful to the anonymous reviewers of the conference version who pointed out related results in the context of resource constrained scheduling problems. The first and third authors were supported in part by NSF grant CCF-0514082.

References

[1] J. Augustine, S. Seiden, Linear time approximation schemes for vehicle scheduling problems, Theoret. Comput. Sci. 324 (2–3) (2004) 147–160.
[2] B.S. Baker, D.J. Brown, H.P. Katseff, A 5/4 algorithm for two-dimensional packing., J. Algorithms 2 (4) (1981) 348–368.
[3] B.S. Baker, E.G. Coffman, R.L. Rivest, Orthogonal packings in two dimensions., SIAM J. Comput. 9 (4) (1980) 846–855.
[4] S. Banerjee, E. Bozorgzadeh, N. Dutt, Physically-aware hw-sw partitioning for reconfigurable architectures with partial dynamic reconfiguration, in: Design Automation Conference, 2005, pp. 335–340.
[5] A.L. Buchsbaum, H. Karloff, C. Kenyon, N. Reingold, M. Thorup, Opt versus load in dynamic storage allocation, SIAM J. Comput. 33 (3) (2004) 632–646.
[6] E.G. Coffman, M.R. Garey, D.S. Johnson, R.E. Tarjan, Performance bounds for level-oriented two-dimensional packing algorithms., SIAM J. Comput. 9 (4) (1980) 808–826.
[7] J. Csirik, G.J. Woeginger, Shelf algorithms for on-line strip packing, Inform. Process. Lett. 63 (4) (1997) 171–175.
[8] W.F. de la Vega, G.S. Lueker, Bin packing can be solved within $1 + \epsilon$ in linear time, Combinatorica 1 (4) (1981) 349–355.
[9] W.F. de la Vega, V. Zissimopoulos, An approximation scheme for strip packing of rectangles with bounded dimensions., Discrete Appl. Math. 82 (1–3) (1998) 93–101.
[10] M. Grotschel, L. Lovasz, A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, Combinatorica 1 (1981) 169–197.
[11] K. Jansen, R. van Stee, On strip packing with rotations, in: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing, ACM Press, New York, NY, USA, 2005, pp. 755–761.
[12] K. Jansen, H. Zhang, Scheduling malleable tasks with precedence constraints, in: Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures, ACM Press, New York, NY, USA, 2005, pp. 86–95.
[13] D.S. Johnson, M.R. Garey, R.L. Graham, A.C. Yao, Resource constrained scheduling as generalized bin packing, J. Combin. Theory 21 (3) (1976) 257–298.
[14] N. Karmarkar, A new polynomial-time algorithm for linear programming, Combinatorica 4 (1984) 373–395.
[15] N. Karmarkar, R. Karp, An efficient approximation scheme for the one dimensional bin-packing problem, in: Proceedings of the 23rd Sympos. Foundations Computer Sci., FOCS, Tucson, AZ, 1982, pp. 312–320.
[16] C. Kenyon, E. Rémila, A near-optimal solution to a two-dimensional cutting stock problem, Math. Oper. Res. 25 (4) (2000) 645–656.
[17] R. Lepère, D. Trystram, G.J. Woeginger, Approximation algorithms for scheduling malleable tasks under precedence constraints, in: Proceedings of the 9th Annual European Symposium on Algorithms, Springer-Verlag, London, UK, 2001, pp. 146–157.
[18] C. Martel, Preemptive scheduling with release times, deadlines, and due times, J. Assoc. Comput. Mach. 29 (3) (1982) 812–829.
[19] S. Martello, M. Monaci, D. Vigo, An exact approach to the strip-packing problem, INFORMS J. Comput. 15 (3) (2003) 310–319.
[20] G. Mounie, C. Rapine, D. Trystram, Efficient approximation algorithms for scheduling malleable tasks, in: Proceedings of the Eleventh Annual ACM Symposium on Parallel Algorithms and Architectures, ACM Press, New York, NY, USA, 1999, pp. 23–32.
[21] M. Queyranne, Bounds for assembly line balancing heuristics, Oper. Res. 33 (6) (1985) 1353–1359.
[22] I. Schiermeyer, Reverse-fit: A 2-optimal algorithm for packing rectangles, in: Proceedings of the Second Annual European Symposium on Algorithms, Springer-Verlag, London, UK, 1994, pp. 290–299.
[23] C. Steiger, H. Walder, M. Platzner, Operating systems for reconfigurable embedded platforms: Online scheduling of real-time tasks, IEEE Trans. Comput. 53 (11) (2004) 1393–1407.
[24] A. Steinberg, A strip-packing algorithm with absolute performance bound 2, SIAM J. Comput. 26 (2) (1997) 401–409.
[25] Xilinx. http://www.xilinx.com.