Strip Packing with Precedence Constraints and Strip Packing with Release Times

John Augustine
jea@ics.uci.edu

Sudarshan Banerjee
banerjee@ics.uci.edu

Sandy Irani
irani@ics.uci.edu

Donald Bren School of Information and Computer Sciences
University of California, Irvine
Irvine, CA 92697

ABSTRACT

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 can be used to model scheduling problems in which tasks require a contiguous subset of identical resources that are arranged in a linear topology. The particular variants studied here are motivated by scheduling tasks for dynamically reconfigurable Field-Programmable Gate Arrays (FPGAs) comprised of an array of computing columns. Each column is a computing resource and the array of columns forms the linear topology of resources. We assume that the given FPGA has K columns, where K is a fixed positive integer, and each task occupies a contiguous subset of these columns. For the case in which tasks have precedence constraints, we give an $O(\log n)$ approximation, where n is the number of tasks. 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 $(1+\epsilon)$-approximation scheme. We make the standard assumption that the rectangles have height at most 1. In addition, we also require widths to be in $[\frac{1}{K}, 1]$, i.e., the rectangles are at least as wide as a column in the FPGA. Our running time is polynomial in n and $1/\epsilon$, but exponential in K.

Categories and Subject Descriptors

F.2.0 [**Analysis Of Algorithms And Problem Complexity**]: Nonnumerical Algorithms and Problems—*Sequencing and Scheduling*

General Terms

Algorithms

SPAA'06, July 30–August 2, 2006, Cambridge, Massachusetts, USA.

Keywords

Strip Packing, Reconfigurable Architectures, Precedence Constraints, Release Times, Scheduling

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 so that the height of the area 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 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* interchangably 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 [1] additionally allow the device to be con-

figured *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 [5] leads to our first problem of *Strip packing with precedence constraints*. We give an approximation algorithm for this problem that is within a factor of $\log n + 2$ of optimal. 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 sophistocated 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 [24]. 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 lease 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. The techniques developed to solve bin-packing have had considerable impact on our understanding of linear programming. In the early 80s, Fernandez de la Vega and Lueker [9] 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. Along with bin packing, strip packing was also studied extensively [4, 7, 3] 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 [23] and Steinberg [25]. The asymptotic performance has been impressive with the first asymptotic PTAS by Fernandez de la Vega and Zissimopoulos [10] 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 [7, 9, 15, 10]. A lot of the original strip packing was motivated by stock cutting which did not allow rectangle rotations because the materials often had designs in the material that required a certain orientation. However, motivated by applications of strip packing to scheduling, researchers have started studying several variants such as allowing 90-degree rotations [12], online versions of strip packing [8], exact approaches to strip packing [20] and malleability in rectangles [21]. The storage allocation problem studied by Buchsbaum et al. [6] 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 [19, 17, 5, 24, 13]. Of particular note are precedence constrained bin-packing problem variants [18, 22] 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 constrained strip packing.

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 $F(s) = 0$.
- If $\mathcal{IN}(s) \neq \emptyset$, $F(s) = \max_{s' \in \mathcal{IN}(s)}(F(s') + h_s)$.

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 maximum height of any rectangle in S' under this placement:

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

In particular, we will require that $\mathcal{A}$ has the property that for any set of rectangles S' that have no precedence constraints:

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

The property is satisfied by the algorithms given in [25] and [23].

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 $(S, 0)$.

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: $y_{bot} = \mathcal{DC}(y, S_{bot})$
9: Place rectangles in S_{mid} according to $\mathcal{A}(y_{bot}, S_{mid})$
10: Assign: $y_{mid} = \mathcal{A}(y_{bot}, S_{mid})$
11: Place rectangles in S_{top} according to $\mathcal{DC}(y_{mid}, S_{top})$
12: Return $\mathcal{DC}(y_{mid}, S_{top})$

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 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. Suppose there were two tasks s and s' in S' such that there is a path from s to s' in the dependency graph. Let s_{pred} be the predecessor of s' along this path. Since (s_{pred}, s') is an edge in E, it must be the case that $F(s') \geq F(s_{pred}) + h_{s'}$. Furthermore, the values of F along a path can not decrease, so it must be the case that $F(s') \geq F(s) + h_{s'}$. However, by the definition of S', it must be the case that

$$F(s) > y \geq F(s') - h_{s'}$$

which implies that $F(s) + h_{s'} > F(s')$, a contradiction. □

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

LEMMA 2. *The set S_{mid} can not be empty.*

PROOF. Suppose for a contradiction that S_{mid} is empty. Consider the DAG induced by the rectangles in S_{top}. Pick any rectangle s in this subgraph that has zero in-degree. All of the rectangles with edges into s in the original graph must be in S_{bot}. By definition of S_{bot}, $\max_{s \in \mathcal{IN}(s)} F(s) \leq H/2$. Therefore $F(s) \leq H/2 + h_s$ which implies that s should not be in S_{top}. □

Now we can establish the bound on the height of the schedule produced by DC.

THEOREM 3. $\mathcal{DC}(S) \leq (\log n + 2)OPT(S)$.

PROOF. We will show that $\mathcal{DC}(S) \leq \log n \cdot F(S) + 2 \cdot \mathcal{ARE}\mathcal{A}(S)$. 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, we know that $n_{top} + n_{bot} < n$. By induction, we know that

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

and

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

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

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

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

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

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

and

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

.

PROOF. Without loss of generality, we assume $n = 2^{k+2} - 2$ 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 Figure 1. Each rectangle $s \in S_{wide}$, shown as unshaded rectangles in Figure 1, has $h_s = \epsilon \to 0$ and

$w_s = 1$. The tall rectangles are the shaded rectangles in Figure 1. S_{tall} is sorted in non-increasing order of heights and $S_{tall}(i) = 1/(2^{\lfloor \log i \rfloor})$, $1 \leq i \leq n/2$. Notice that there are k clusters of heights. All rectangles in S_{tall} have a width of $1/k$. The precedence constraints are as illustrated in Figure 1. I.e., there are k chains in $G(S, E)$ and each chain i is comprised of a sequence $S^i_{tall} = (s : s \in S_{tall} \bigwedge h_s = 1/2^{i-1})$ (in some arbitrary order) , where $1 \leq i \leq k$. 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 $\max_{s \in S} F(s) = \mathcal{ARE A}(S) = 1$, 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 because 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 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 $k/2 \in \Omega(\log n)$. □

2.2 Fixed height

In this section, we study precedence constrained strip packing when the heights of the rectangles are fixed at some value h. 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 [18] 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 duration in $(0, 1]$ associated with it. We also have a sequence of bins and each bin can hold tasks that have a total duration 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. We show that it carries over to our problem. We then provide an absolute 3-approximation algorithm $\mathcal{F}$ for our problem. 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 any solution S of height H for an instance of the strip packing problem with precedence constraints and uniform rectangle heights. Define a shelf i for some positive integer i to be the portion of the strip from height $(i-1)h$ to ih. We can easily construct a solution $S^*$of height $H^* \leq H$ such that each rectangle fits in a shelf. We iteratively pick the rectangle s that spans two shelves and has the lowest x_s and slide it down till 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' will also span 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 duration w, our problem reduces to the precedence constrained bin packing problem. Therefore, their asymptotic results apply to our problem also.

The algorithm $\mathcal{F}$ maintains a ready list of rectangles such that all their predecessors in $G(S, E)$ have already been scheduled. We use a shelf algorithm wherein the rectangles from the ready list are placed from left to right in a shelf. When there is room in the topmost shelf for any arbitrary rectangle in the ready list, we place it. We continue to do so until no rectangle fits in this shelf. We then close the top shelf and open a new shelf. Every time we open a new shelf, we repopulate the ready list (as new rectangles may have been made available) and continue fitting rectangles into the new shelf. If the ready list is empty, but we still have unscheduled rectangles, we close the current shelf and open a new one (causing the ready list to repopulate). We call this a *skip*. We repeat this until we exhaust all of rectangles. The following lemma bounds the number of skips.

LEMMA 5. *The number of skips performed by $\mathcal{F}$ is at most $OPT(S, E)/h$.*

PROOF. First observe that if there is path of length p in $G = (S, E)$, then $p \cdot h$ 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 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 h, 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. This is true because if a green shelf did not cause a skip step, the contents of shelf $i+1$ could have been shifted down to fit in shelf i. The total height of the packing is $(r+g)h$. Since red shelves have density at least 1/2, $rh \leq 2\mathcal{ARE A}(S) \leq 2 \cdot OPT(S, E)$. We also have that $gh \leq OPT(S, E)$ by Lemma 5. Therefore, $(r+g)h \leq 3 \cdot OPT(S, E)$. □

Similar to Lemma 4, Lemma 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 7. *For all $n = 3k$, where k is a positive integer, there exist instances of the strip packing problem with precedence constraints and uniform height such that*

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

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 Figure 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 Figure 2, of height 1 and width $1/2+\epsilon$. The precedence constraints are as shown in the figure. 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. Each rectangle is treated as an indivisible unit and therefore cannot be broken into smaller pieces in any fashion. As usual, we are trying to minimize the maximum height given by $\max_{s \in S} y_s + h_s = OPT(S)$, 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 denote a general instance of this problem as P and the restricted instance with widths in $[\frac{1}{K}, 1]$ is $P(w \geq \frac{1}{K})$. We provide an asymptotic PTAS for $P(w \geq \frac{1}{K})$. Our approach is to achieve results for versions of P that are simplified in various ways in succession and then show how these simplifying assumptions can be removed. For instance, in fractional packing, we relax the indivisibility constraint. In other words, we are allowed to use horizontal cuts on the rectangles as long as the total height of the rectangles of a particular width and release time are maintained as in the original problem. We denote problem instances with various simplifications by appropriate parameters as shown in Table 2.2.

We abuse the notation slightly and use the solution S and its variants to refer both to the algorithm and to the height of the packing produced by the algorithm on the corresponding problem instances P and its variants.

We first provide a polynomial time approximation scheme $S(f, W, w \geq \frac{1}{K}, R)$ for a problem instance $P(f, W, w \geq \frac{1}{K}, R)$ restricted to a fixed number of widths and release times and then show how to convert the solution to the integral version while only paying an increase in the schedule height that is bounded by a constant. W is the number of allowable widths bounded by the range $[\frac{1}{K}, 1]$ and R is the number of positive release times, where W and R are positive integer constants. The widths and release times (sorted in increasing order) are denoted by ω_i, $1 \leq i \leq W$ and $\rho_j > 0$, $1 \leq j \leq R$ respectively. 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. Without loss of generality, we can assume that $\rho_0 = 0$ and for ease of notation, we define an extra $\rho_{R+1} = \infty$. Define phase j to be the time between ρ_{j-1} to ρ_j. Notice that there are $R + 1$ phases in all. We define vector $B_j = (b_j^1, b_j^2, \cdots, b_j^W)$ such that

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

We now introduce the concept of *configurations* that was initially employed in the context of bin-packing [9] and subsequently been used in strip packing as well [9, 10, 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, of all possible configurations that can be formed using the allowed widths. Intuitively, each configuration is the possible combination of widths that can be contained within the 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. Notice also that the configurations do not control the exact ordering of the widths within them, but we will see that these two aspects don't hurt the asymptotic approximation scheme that we provide. 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 $1 \leq j \leq R + 1$. 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: $X_j = (x_j^1, x_j^2, \cdots, x_j^Q)$. 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 $R + 1$. Hence, our objective function is

$$\min \sum_q x_{R+1}^q \tag{1}$$

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 $r_j - r_{j-1}$, i.e., we cannot pack more than is allowed in each phase. Therefore, we can write the *packing constraints* as follows for every integer $j \in [1, R + 1]$:

$$\sum_q x_j^q \leq r_j - r_{j-1}. \tag{2}$$

Simplifications	Problem Instance	Solution
None	P	S
Rectangle widths are bounded within $[\frac{1}{K}, 1]$	$P(w \geq \frac{1}{K})$	$S(w \geq \frac{1}{K})$
Fractional packing is allowed and widths are bounded within $[\frac{1}{K}, 1]$	$P(f, w \geq \frac{1}{K})$	$S(f, w \geq \frac{1}{K})$
Fractional packing is allowed with widths restricted to $[\frac{1}{K}, 1]$ and fixed number of release times R	$P(f, w \geq \frac{1}{K}, R)$	$S(f, w \geq \frac{1}{K}, R)$
Fractional packing is allowed and fixed number of widths that are bounded within $[\frac{1}{K}, 1]$ and fixed number of release times R	$P(f, W, w \geq \frac{1}{K}, R)$	$S(f, W, w \geq \frac{1}{K}, R)$
Integral packing of rectangles and fixed number of widths that are bounded within $[\frac{1}{K}, 1]$ and fixed number of release times R	$P(W, w \geq \frac{1}{K}, R)$	$S(W, w \geq \frac{1}{K}, R)$

Table 1: Notations for problem instances at various levels of simplification

Now, we finally 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 r_j should be accounted for in the linear programming constraint as well. 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 $\sum_q a_{iq} \cdot x_j^q = A \cdot X_j$ 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 $R+1$ is at least the sum of the heights of the rectangles of width ω_i whose release time happens on or after r_{k-1}. We need that for each $1 \leq k \leq R+1$:

$$\sum_{j=k}^{R+1} A \cdot X_j \geq \sum_{j=k}^{R+1} B_{j-1}. \tag{3}$$

We denote the above linear programming formulation consisting of the objective function in (1) and constraints in (2) and (3) along with the non-negativity constraint on the variables as $\mathcal{LP}$. We provide the following lemma relating $\mathcal{LP}$ to $P(f, W, w \geq \frac{1}{K}, R)$, but defer its proof to the Appendix.

LEMMA 8. *The sum of the optimal solution to the linear programming formulation $\mathcal{LP}$ and ρ_R will be the height of the optimal packing for $P(f, W, w \geq \frac{1}{K}, R)$. In addition, any feasible solution $S(\mathcal{LP})$to $\mathcal{LP}$ can be used to construct a solution to the fractional packing problem of height $\rho_R + S(\mathcal{LP})$. Furthermore, the optimal solution to $P(f, W, w \geq \frac{1}{K}, R)$ uses only $(W+1)(R+1)$ distinct configurations.*

The following lemma 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 configurations used. Since the number of configurations used is bounded by $(W+1)(R+1)$ (by Lemma 8), this will result in an additive increase of $(W+1)(R+1)$ in the height of the final integral solution we obtain.

LEMMA 9. *If a solution $S(f, w \geq \frac{1}{K})$ for an instance $P(f, w \geq \frac{1}{K})$ uses at most k configurations, then it can be converted to an integral solution $S(w \geq \frac{1}{K})$ to the same problem instance such that*

$$S(w \geq \frac{1}{K}) \leq S(f, w \geq \frac{1}{K}) + k.$$

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 10. *Given a constant positive integer W, a problem instance $P(w \geq \frac{1}{K}, R)$ can be reduced in polynomial time to an instance $P(W, w \geq \frac{1}{K}, R)$ such that*

$$OPT(f, W, w \geq \frac{1}{K}, R) \leq OPT(f, w \geq \frac{1}{K}, R) \cdot \left(1 + \frac{(R+1)K}{W})\right).$$

The next lemma allows us to reduce our original problem to one in which the number of release times is bounded.

LEMMA 11. *For a fixed $\epsilon_r > 0$, problem instance $P(w \geq \frac{1}{K})$ can be reduced to an instance $P(w \geq \frac{1}{K}, R = 1/\epsilon_r)$ such that $OPT(w \geq \frac{1}{K}, R = 1/\epsilon_r) \leq (1+\epsilon_r)OPT(w \geq \frac{1}{K})$.*

We first show how the various lemmas are linked to give an APTAS for $P(w \geq \frac{1}{K})$. Algorithm 2 outlines the steps.

The output in line 2 of Algorithm 2 outputs a packing for $P(W, w \geq \frac{1}{K}, R)$, but this can be easily adapted to $P(w \geq \frac{1}{K})$ because the release times are higher and the widths are wider than needed.

THEOREM 12. *Algorithm 2 is an asymptotically $(1+\epsilon)$-approximate algorithm for $P(w \geq \frac{1}{K})$ and runs in time polynomial in n, $1/\epsilon$.*

PROOF. It is obvious that the running time is polynomial in n, $1/\epsilon$. By Lemma 8, the solution to the fractional problem obtained in Step 2 uses at most $(W+1)(R+1)$

Algorithm 2 APTAS for $P(f, W, w \geq \frac{1}{K}, R)$

1: **INPUT:** An instance $P(w \geq \frac{1}{K})$ and error parameter ϵ
2: Assign: $\epsilon' := \sqrt{\epsilon/3}$
3: Assign: $R := 1/\epsilon'$
4: Assign: $W := (R+1)/(\epsilon'/K)$
5: **Fractionalize** $P(w \geq \frac{1}{K})$ to $P(f, w \geq \frac{1}{K})$
6: **Reduce** $P(f, w \geq \frac{1}{K})$ to $P(f, w \geq \frac{1}{K}, R)$ according to Lemma 11
7: **Reduce** $P(f, w \geq \frac{1}{K}, R = 1/\epsilon_r)$ to $P(f, W, w \geq \frac{1}{K}, R)$ according to Lemma 10
8: **Solve** $P(f, W, w \geq \frac{1}{K}, R)$ to get $OPT(f, W, w \geq \frac{1}{K}, R)$ as outlined in Lemma 8
9: **Convert** fractional solution $S(f, W, w \geq \frac{1}{K}, R)$ to integral solution $S(W, w \geq \frac{1}{K}, R)$ according to Lemma 9
10: **OUTPUT:** $S(W, w \geq \frac{1}{K}, R)$

configurations. We know

$$\begin{aligned} OPT(f, W, w \geq \frac{1}{K}, R) &\leq (1 + \frac{K \cdot R}{W}) \cdot \\ & \quad OPT(f, w \geq \frac{1}{K}, R) \\ & \quad \{\text{from Lemma 10 }\} \\ OPT(f, w \geq \frac{1}{K}, R) &\leq (1 + \epsilon') \cdot \\ & \quad OPT(f, w \geq \frac{1}{K}) \\ & \quad \{\text{from Lemma 11 }\}. \end{aligned}$$

Therefore,

$$\begin{aligned} OPT(f, W, w \geq \frac{1}{K}, R) &\leq (1 + \frac{K \cdot R}{W})(1 + \epsilon') \cdot \\ & \quad OPT(f, w \geq \frac{1}{K}). \end{aligned}$$

Applying $\epsilon' = \sqrt[3]{\epsilon}$, $R = 1/\epsilon'$, and $W = (R+1)/(\epsilon'/K)$, we get

$$OPT(f, W, w \geq \frac{1}{K}, R) \leq (1 + \epsilon) OPT(f, w \geq \frac{1}{K}).$$

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 9, we get

$$\begin{aligned} S(W, w \geq \frac{1}{K}, R) &\leq (1 + \epsilon) OPT(f, w \geq \frac{1}{K}) \\ &+ (W + 1)(R + 1). \end{aligned}$$

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

Proof of Lemma 8: A similar flavor of linear programming formulation without the packing constraints has been employed in the past by several researchers working on related problems [9, 15, 16]. Given an optimal solution to $\mathcal{LP}$, we simply construct a solution to $P(f, W, w \geq \frac{1}{K}, R)$ 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 solution to $P(f, W, w \geq \frac{1}{K}, R)$ uses only $(W+1)(R+1)$ distinct configurations. Note that techniques in [11, 14] can be used to solve $\mathcal{LP}$, and since they provide optimal solution, the number of non-zero variables that they produce will be at most $(W + 1)(R + 1)$. □

Proof of Lemma 9: Notice that the problem definitions are identical for the two instances except that one is fractional while the other is integral. Hence, we can use $S(f, w \geq \frac{1}{K})$ to provide us a solution $\bar{X}_1, \bar{X}_2, \cdots, \bar{X}_R$, where each $\bar{X}_j$ is associated with a phase j and the number of configurations with non-zero height is at most k. We start with phase 1 and iterate through each configuration in it. we start at height zero and allocate up to height $\bar{x}_1^1 + 1$ to the first configuration in phase j. We successively allocate a height of $\bar{x}_j^q$ for each configuration q, $1 \leq q \leq Q$. We stack these configurations one on top of another. The need for the extra 1 unit of height added to each configuration will be apparent shortly. We now greedily place rectangles from the integral packing $P(w \geq \frac{1}{K})$ into the configurations in the following manner. For each width i, we maintain a list L_i of rectangles of width i from $P(w \geq \frac{1}{K})$ sorted in the non-decreasing order of their release times. We then scan from bottom up and place these rectangles in columns that they fit snug. In the event that there aren't any more rectangles in L_i that are released in the current or prior phases, we move upward to the base of the next phase and continue this process until all rectangles have been exhausted. We can guarantee that all the rectangles from $P(w \geq \frac{1}{K})$ can be placed in the strip in this manner. We can prove this via contradiction. Suppose there is a rectangle s with width w_s that did not fit. This can only happen when all the occurrences of this width in $S(f, w \geq \frac{1}{K})$ are exhausted, i.e., in the last (topmost) phase, say j, where this width occurs. Let $j' \leq j$ be the topmost phase such that the first rectangle of same width as s placed in j' was also released at $r_{(j'-1)}$. Our only concern will be the configurations in $[j', j]$. This range has the property that there was no space wasted in waiting for the right release time before a rectangle of width same as s could be placed. Note that each configuration that has the width w_s represented can be visualized as having one or more columns dedicated for width w_s depending on the number of occurrences of w_s in that configuration. Since each task has a height at most 1, these columns are filled up except for the topmost 1 unit height where a rectangle could not fit. This is precisely the reason for allocating the one extra unit of height per configuration. Given this tight filling of the space in the columns, the inability to fit rectangle s can only be caused because the fractional packing did not allocate enough space for width w_s in the phase range $[j', j]$, which we know is not true. Hence we see a contradiction. Finally, since there are at most k configurations and we increase their heights by at most 1 each, thereby proving the lemma. □

Proof of Lemma 10: This proof uses the grouping technique previously employed in [9, 10, 16], but deviates considerably to account for the R release times. We first outline the reduction and then show that the lemma holds for it. 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 from bottom to top. We call this the *stacking of* S. We define a partial order on sets of input rectangles, $S \leq S'$ when the stacking of S is completely contained within the stacking of S' and all rectangles in S and S' have the same release time. If the sets S and S' have several release times, $S \leq S'$ if for each release time ρ_i, the stacking of $S_i \leq S'_i$, where $S_i = \{s : s \in S \bigwedge r_s = \rho_i\}$ and S'_i is defined likewise.

Let S be the set of input rectangles in $P(w \geq \frac{1}{K}, R)$. We construct a set S^{sup} with at most W distinct widths such that $S \leq S^{sup}$. We partition $S = S_0 \bigcup S_1 \bigcup, \cdots, \bigcup S_R$, where each $S_i = \{s : s \in S \bigwedge r_s = \rho_i\}$, $0 \leq i \leq R$. We stack the rectangles in S_i in non-increasing order from bottom to top in a left justified manner. We denote the height of this stack as $H(S_i)$. We use $W/(R+1)$ out of the W widths per partition S_i of S. Hence, we divide this stack into $W/(R+1)$ equal pieces by cutting the stack with lines $y = \ell H(S_i)(R+1)/W$, where $0 \leq \ell \leq W/(R+1)$. A rectangle is a threshold rectangle if a line $\ell H(S_i)(R+1)/W$ cuts through its interior or aligns with its base. The threshold rectangles partition the stacking of S_i into groups, i.e., each group starts with a threshold rectangle in the stack and includes all rectangles until but not including the next threshold rectangle . To construct S_i^{sup}, we set the widths of all the rectangles in a group to that of its threshold rectangle. For the rectangles in the first group, we round the width to 1. The rest of the attributes of each rectangle such as its height and release time remain unaltered. This new set of rectangles created in this manner will be our S_i^{sup}. We construct $S^{sup} = \bigcup_i S_i^{sup}$. We now show that when we use the rectangles of S^{sup} to construct $P(W, w \geq \frac{1}{K}, R)$, the lemma will hold.

In order to prove this lemma, we consider two other sets of rectangles $S^{inf} = \bigcup_i S_i^{inf}$ and $S' = \bigcup_i S'_i$. Similar to constructing S_i^{sup}, each $S_i^{\inf}$ and S'_i are constructed from the groupings as explained previously. Each the $S_i^{\inf}$, of cardinality $R+1$ is a set of rectangles of height $H(S_i)(R+1)/W$ associated with each group. Each rectangle in $S_i^{\inf}$ has the width of the threshold rectangle in the group above that of itself. The rectangle of the topmost group is assigned a width of 0. The rectangles in S'_i, where $|S'_i| = W/(R+1)$ as well, are also of height $H(S_i)(R+1)/W$ and each rectangle is associated with a group. Each rectangle in S'_i has width equal to that of its own threshold rectangle. However, the width of the rectangle associated with the first group is 1. We can see that $S^{inf} \leq S \leq S^{sup} \leq S'$. Note that in the case of S^{inf} and S', the portions of the stack from $\ell H(S_i)R/W$ to $(\ell+1)H(S_i)R/W$, $0 \leq \ell \leq (W/R) - 1$ have the same width, while this may not be the case for S^{sup}. The fractional strip packing formulations using S^{inf} and S' are almost the same except that S' has R extra rectangles of height $H(S_i)(R+1)/W$ and width 1, corresponding to each release time ρ_i. We denote these packings as $S^{inf}(f, W, w \geq \frac{1}{K}, R)$ and $S'(f, W, w \geq \frac{1}{K}, R)$ respectively. The optimal fractional strip packing schedules for the two differ by exactly $\sum_i H(S_i)(R+1)/W = H(S)(R+1)/W$. Therefore,

$$\begin{aligned} S'(f, W, w \geq \frac{1}{K}, R) &= S^{inf}(f, W, w \geq \frac{1}{K}, R) \\ &+ H(S)(R+1)/W. \end{aligned} \quad (4)$$

Since the widths of the rectangles are lower bounded by ϵ, $H(S) \cdot \epsilon \leq A(S) \leq S(f, W, w \geq \frac{1}{K}, R)$. Also, for any two sets of rectangles S'' and S''' such that $S'' \leq S'''$, we know that $S''(f, w \geq \frac{1}{K}, R) \leq S'''(f, w \geq \frac{1}{K}, R)$. Therefore, we have

$$\begin{aligned} S^{inf}(f, W, w \geq \frac{1}{K}, R) &\leq S(f, w \geq \frac{1}{K}, R) \\ &\leq S^{sup}(f, W, w \geq \frac{1}{K}, R) \\ &\leq S'(f, W, w \geq \frac{1}{K}, R). \end{aligned} \quad (5)$$

From Equations (4) and (5), we get

$$\begin{aligned} S^{sup}(f, W, w \geq \frac{1}{K}, R) &\leq S(f, w \geq \frac{1}{K}, R) \\ &(1 + \frac{R+1}{W/K}), \end{aligned}$$

which holds for OPT as well. Assigning

$$P(f, W, w \geq \frac{1}{K}, R) \leftarrow P^{sup}(f, W, w \geq \frac{1}{K}, R)$$

proves our theorem. □

Proof of Lemma 11: We provide a constructive proof that bounds the number of release times similar to [2]. Define $r_{\max} = \max_{s \in S} r_s$. Note that $r_{\max}$ is a lower bound for both $S(w \geq \frac{1}{K})$ and $S(w \geq \frac{1}{K}, R = 1/\epsilon_r)$. Define $\delta = \epsilon_r r_{\max}$. We define $\rho_j = j \cdot \delta$, for $1 \leq j \leq 1/\epsilon_r$. Notice that this creates $1/\epsilon_r$ release times. We now define two problem instances for $P(w \geq \frac{1}{K})$:

$P^{\downarrow}(w \geq \frac{1}{K})$: For each rectangle $s \in S$ defined in the problem instance $P(w \geq \frac{1}{K})$, we have a corresponding rectangle $s^{\downarrow}$ in $P^{\downarrow}(w \geq \frac{1}{K})$ with the same dimensions as in s, but its release time is $\max_{1 \leq i \leq n} \rho_i < r_s$. Also, define the optimal packing for this problem to be $OPT^{\downarrow}(w \geq \frac{1}{K})$.

$P^{\uparrow}(w \geq \frac{1}{K})$: For each rectangle $s \in S$ defined in the problem instance $P(w \geq \frac{1}{K})$, we have a corresponding rectangle $s^{\uparrow}$ in $P^{\uparrow}(w \geq \frac{1}{K})$ with the same dimensions as in s, but its release time is $\min_{1 \leq i \leq n} \rho_i \geq r_s$. Also, define the optimal packing for this problem to be $OPT^{\uparrow}(w \geq \frac{1}{K})$.

Notice that there is an important structural relationship that exists between the two problem instances. For every rectangle $s^{\downarrow}$ in $P^{\downarrow}(w \geq \frac{1}{K})$, there is a corresponding $s^{\uparrow}$ in $P^{\uparrow}(w \geq \frac{1}{K})$ whose release time is exactly $r_{s^{\downarrow}} + \delta$. Therefore, if there is a strip packing for $P^{\downarrow}(w \geq \frac{1}{K})$ with height $H^{\downarrow}$, then there is a strip packing for $P^{\uparrow}(w \geq \frac{1}{K})$ with height $H^{\uparrow} = H^{\downarrow} + \delta$. This is true because we can just push all the rectangles in $P^{\downarrow}(w \geq \frac{1}{K})$ up by a distance δ in order to satisfy the release time constraints for $P^{\uparrow}(w \geq \frac{1}{K})$. Hence we can state that

$$OPT^{\uparrow}(w \geq \frac{1}{K}) = OPT^{\downarrow}(w \geq \frac{1}{K}) + \delta. \quad (6)$$

In addition, each rectangle s in $P(w \geq \frac{1}{K})$ has release times sandwiched in $[r_{s^{\downarrow}}, r_{s^{\uparrow}}]$. Hence we can state that

$$OPT^{\downarrow}(w \geq \frac{1}{K}) \leq OPT(w \geq \frac{1}{K}) \leq OPT^{\uparrow}(w \geq \frac{1}{K}). \quad (7)$$

Hence,

$$\begin{aligned} OPT^{\uparrow}(w \geq \frac{1}{K}) &\leq OPT(w \geq \frac{1}{K}) + \delta \\ &= OPT(w \geq \frac{1}{K}) + \epsilon_r r_{\max} \\ &\leq (1+\epsilon_r) OPT(w \geq \frac{1}{K}). \end{aligned}$$

Hence, $P^{\uparrow}(w \geq \frac{1}{K})$ serves as the reduced problem instance that has exactly R release times. □

4. ACKNOWLEDGMENTS

The authors are very grateful to Dr. George Lueker for providing us some key insights. We are also thankful to the anonymous reviewers who pointed out related results in the context of resource constrained scheduling problems.

5. REFERENCES

[1] Xilinx. `http://www.xilinx.com`.

[2] J. Augustine and S. Seiden. Linear time approximation schemes for vehicle scheduling problems. *Theoretical Computer Science*, 324(2-3):147–160, 2004.

[3] B. S. Baker, D. J. Brown, and H. P. Katseff. A 5/4 algorithm for two-dimensional packing. *J. Algorithms*, 2(4):348–368, 1981.

[4] B. S. Baker, E. G. Coffman, and R. L. Rivest. Orthogonal packings in two dimensions. *SIAM J. Comput.*, 9(4):846–855, 1980.

[5] S. Banerjee, E. Bozorgzadeh, and N. Dutt. Physically-aware hw-sw partitioning for reconfigurable architectures with partial dynamic reconfiguration. In *Design Automation Conference*, pages 335–340, 2005.

[6] A. L. Buchsbaum, H. Karloff, C. Kenyon, N. Reingold, and M. Thorup. Opt versus load in dynamic storage allocation. *SIAM J. Comput.*, 33(3):632–646, 2004.

[7] E. G. Coffman, M. R. Garey, D. S. Johnson, and R. E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM J. Comput.*, 9(4):808–826, 1980.

[8] J. Csirik and G. J. Woeginger. Shelf algorithms for on-line strip packing. *Inf. Process. Lett.*, 63(4):171–175, 1997.

[9] W. F. de la Vega and G. S. Lueker. Bin packing can be solved within $1+\epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

[10] W. F. de la Vega and V. Zissimopoulos. An approximation scheme for strip packing of rectangles with bounded dimensions. *Discrete Applied Mathematics*, 82(1-3):93–101, 1998.

[11] M. Grotschel, L. Lovasz, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[12] K. Jansen and R. van Stee. On strip packing with rotations. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 755–761, New York, NY, USA, 2005. ACM Press.

[13] K. Jansen and H. Zhang. Scheduling malleable tasks with precedence constraints. In *Proceedings of the 17th annual ACM symposium on Parallelism in algorithms and architectures*, pages 86–95, New York, NY, USA, 2005. ACM Press.

[14] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[15] N. Karmarkar and R. Karp. An efficient approximation scheme for the one dimensional bin-packing problem. *Proceedings of the 23rd Sympos. Foundations Computer Sci. (FOCS), Tucson, AZ*, pages 312–320, 1982.

[16] C. Kenyon and E. Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Math. Oper. Res.*, 25(4):645–656, 2000.

[17] R. Lepère, D. Trystram, and G. J. Woeginger. Approximation algorithms for scheduling malleable tasks under precedence constraints. In *Proceedings of the 9th Annual European Symposium on Algorithms*, pages 146–157, London, UK, 2001. Springer-Verlag.

[18] D. S. J. M. R. Garey, R. L. Graham and A. C. Yao. Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory*, 21(3):257–298, Nov 1976.

[19] C. Martel. Preemptive scheduling with release times, deadlines, and due times. *Journal of the Association of Computing Machinery*, 29(3):812–829, 1982.

[20] S. Martello, M. Monaci, and D. Vigo. An exact approach to the strip-packing problem. *INFORMS J. on Computing*, 15(3):310–319, 2003.

[21] G. Mounie, C. Rapine, and D. Trystram. Efficient approximation algorithms for scheduling malleable tasks. In *Proceedings of the eleventh annual ACM symposium on Parallel algorithms and architectures*, pages 23–32, New York, NY, USA, 1999. ACM Press.

[22] M. Queyranne. Bounds for assembly line balancing heuristics. *Operations Research*, 33(6):1353–1359, 1985.

[23] I. Schiermeyer. Reverse-fit: A 2-optimal algorithm for packing rectangles. In *Proceedings of the Second Annual European Symposium on Algorithms*, pages 290–299, London, UK, 1994. Springer-Verlag.

[24] C. Steiger, H. Walder, and M. Platzner. Operating systems for reconfigurable embedded platforms: Online scheduling of real-time tasks. *IEEE Transactions on Computers*, 53(11):1393–1407, 2004.

[25] A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM J. Comput.*, 26(2):401–409, 1997.

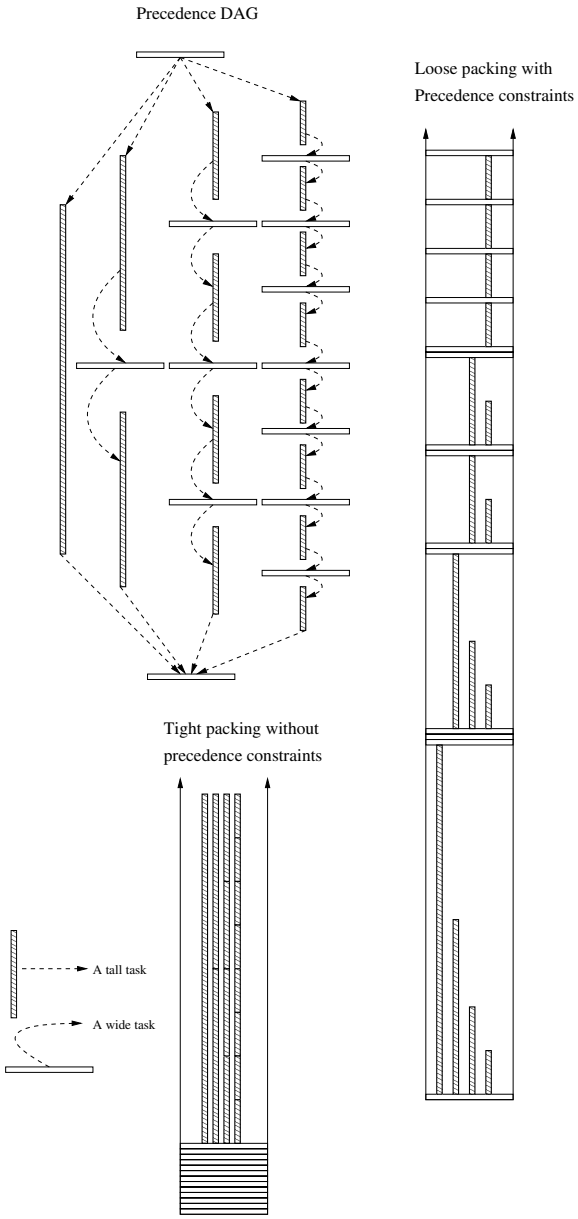

Figure 1: Illustrates the difficulty in proving that an algorithm is $\omega(\log n)$-approximate.

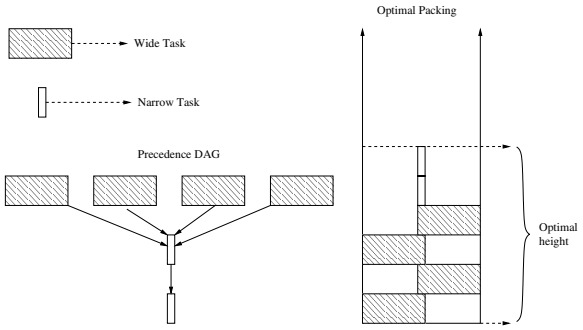

Figure 2: Illustrates the difficulty in proving that an algorithm is $\omega(\log n)$-approximate.