March 2014

High Multiplicity Strip Packing

Devin Price
The University of Western Ontario

Supervisor
Dr. Roberto Solis-Oba
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

HIGH MULTIPLICITY STRIP PACKING
(Thesis format: Monograph)

by

Devin Price

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract

An instance of the two-dimensional strip packing problem is specified by n rectangular items, each having a width, $0 < w_n \leq 1$, and height, $0 < h_n \leq 1$. The objective is to place these items into a strip of width 1, without rotations, such that they are nonoverlapping and the total height of the resulting packing is minimized. In this thesis, we consider the version of the two-dimensional strip packing problem where there is a constant number K of distinct rectangle sizes and present an $OPT + K - 1$ polynomial-time approximation algorithm for it. This beats a previous algorithm with a worst case bound of $OPT + K$; the time complexity of that algorithm was not known and here we show that it runs in polynomial time.

Keywords: strip packing, approximation algorithm, optimization

Contents

List of Figures

Chapter 1

Introduction

An instance of the two-dimensional strip packing problem (2DSPP) is specified by a set R of n rectangles, each rectangle r_i having width, $0 < w_i \leq 1$, and height, $0 < h_i \leq 1$. The objective is to place these rectangles in a strip of width 1 such that they are nonoverlapping and the total height of the resulting packing is minimized. We assume that rectangles have a fixed orientation so their sides are parallel to the sides of the strip and they cannot be rotated. Figure 1.1 shows an instance of the 2DSPP.

The 2DSPP is an optimization problem known to be NP-hard, which means that it is at least as difficult to solve as any problem in the class NP. NP is the computational complexity class consisting of problems for which feasibility of their solutions can be verified in polynomial time. Currently, there is no known algorithm for any NP-hard problem that runs in time polynomial in the size of the input, and it is widely believed that no polynomial time algorithms exist for these problems. Seminal work of Cook [5] proved that a polynomial time algorithm for any NP-hard problem can be transformed into a polynomial time algorithm for any other problem in the class NP-complete, so all these problems are equally "hard" from the point of view of the existence of polynomial time algorithms for them. The class of NP-hard problems is larger than the class NP-complete, as every NP-complete problem is NP-hard and belongs to the class NP. The class P includes those problems which can be solved in polynomial time

using deterministic algorithms. One of the most important questions in computer science is whether or not $P = NP$. Assuming that the classes P and NP are not equal, NP-hard problems cannot be solved in polynomial time by deterministic algorithms. Therefore, approximation algorithms are widely used to deal with this class of problems as they can find near-optimal solutions for NP-hard problems in polynomial time. This is not the same as heuristic-based approaches; approximation algorithms are expected to have provable worst-case runtimes and solution qualities, while for heuristics there is no guarantee on the quality of the solutions that they produce and/or on their running times.

Approximation algorithms often approximate solutions within a constant factor of the optimum. Problems for which such algorithms exists belong to the complexity class APX. For some APX problems, their solutions can be approximated within any constant factor greater than 1 in polynomial time; such algorithms are referred to as *polynomial-time approximation schemes* (PTAS). Additionally, there exist problems in APX for which there is no PTAS; that is, problems that can be approximated within some constant factor, but not every constant factor larger than 1.

For decades geometric packing problems have been a widely studied field of research. Many types of packing problems exist, including bin packing, rectangle packing, and strip packing, among many others. In each of these problems, items with specified sizes and/or weights need to be placed inside of one or more containers to optimize some objective function.

In this thesis we are interested in a version of the strip packing problem known as the high-multiplicity strip packing problem, where the number of distinct rectangle sizes, or "types" of rectangles, is fixed.

1.1 Applications and Related Problems

Packing problems have many real-world and theoretical applications. In this section we mention just a few of these problems. The "simplest" packing problem may be the bin packing

problem. In the bin packing problem, we are given n items where the ith item has weight $w_i \leq 1$. The objective is to place these items inside bins of unit capacity in such a way that as few bins are used as possible. Bin packing has numerous applications, aside from the obvious relation to packing actual boxes into containers.

Imagine a tour bus company was servicing many groups of tourists. Group members do not want to be separated, but the bus company does not want to use more buses than necessary, to maximize profit. Each bus can be considered a bin of capacity equal to its number of seats and each group an item of weight equal to the number of tourists in the group. Solving the corresponding bin packing problem will result in a solution that minimizes the number of buses used while keeping groups unseparated on buses. Bin packing also has important implications to 2DSPP. In fact, bin packing is equivalent to 2DSPP if all rectangles have equal height; therefore, many techniques used in bin packing approximation naturally lend themselves to strip packing as well.

Multiprocessor scheduling is closely related to packing problems. In the multiprocessor scheduling problem, an input consists of a set of jobs each with a length or required processing time. The task is to schedule these jobs on a number of machines such that all jobs are completed in the minimum time. This problem is effectively a bin packing problem where the number of bins is equal to the number of machines and the objective is to pack all items into bins of minimum capacity. In 1978, Coffman et al. [3] presented an algorithm based on bin-packing techniques for the multiprocessor scheduling problem with an approximation ratio of 1.22. This algorithm improved on the previous best bound of $\frac{4}{3}$. Hochbaum and Shmoys presented a PTAS for this problem in 1988 [14].

Another famous geometric problem is the rectangle packing problem. In this problem there is a single rectangular bin into which rectangles are to be placed. The goal is to pack rectangles in the bin so that some objective function is optimized such as the number of rectangles packed or their total area. The rectangle packing problem has applications, for example, in the efficient use of a single time-shared resource (the container rectangle). For the rectangle packing

problem, a PTAS was presented by Fishkin et al. [9] in 2005. This PTAS is based on "dual approximation" techniques, where "better" infeasible solutions are considered and converted to suboptimal feasible solutions. In [9], the problem of solving the machine scheduling problem is reduced to finding a dual approximation algorithm for the bin packing problem.

The strip packing problem is different from the rectangle packing problem in that the height of the strip is not fixed, but minimized while ensuring that all rectangles are packed. One application of the strip packing problem is static scheduling of parallel tasks. Consider a scenario where there are parallel tasks that must be scheduled, each requiring a number of adjacent processors and processor time. Each task in this problem corresponds to a rectangle in a strip packing problem. The width of a rectangle is equal to the number of processors required by the corresponding task and the height is equal to its required processor time. The width of the strip is the number of processors available in the system. An optimal solution to this strip packing problem will give an optimal solution to the original scheduling problem where the height of the final packing corresponds to the total execution time for the tasks.

The knapsack problem is another class of packing problem. In the knapsack problem, there is a single "bin" of given capacity, known as the knapsack, and items have weights and values. The objective is to place items into the knapsack such that their total weight does not exceed the knapsack's capacity and the total value of the items in the knapsack is maximized. Consider the situation where there is a set of tasks that must be completed before a deadline, each having a time requirement and an expected profit if completed by the deadline. This is a knapsack problem where a task corresponds to an item with weight and value equal to the time required and expected profit for the task, respectively. The capacity of the knapsack represents the deadline by which tasks must be completed. Solving this problem optimally would result in the set of tasks that maximizes profit and can be completed before the deadline.

In practice the number of distinct rectangle sizes, or types, involved in a packing problem is often small. When the number of types is constant, the corresponding problem becomes a special type of packing problem; packing problems with a small number of item types are

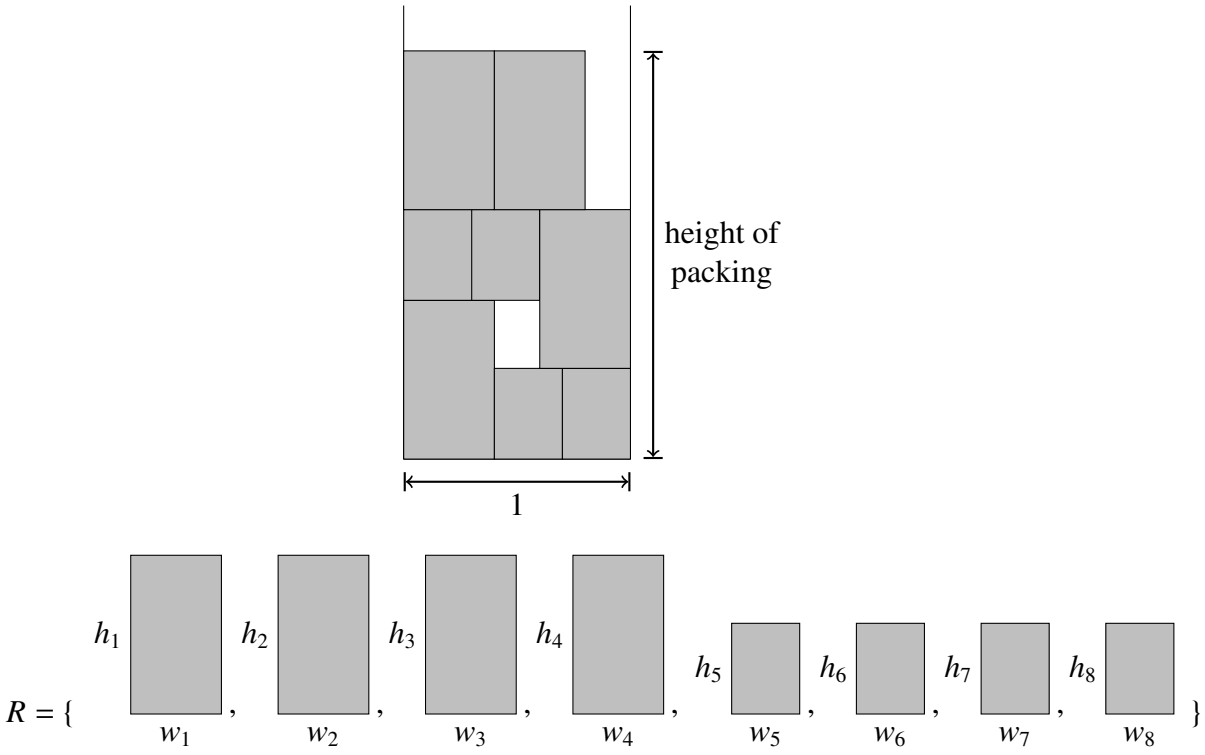

Figure 1.1: An instance of the two-dimensional strip packing problem with 8 rectangles.

called *high multiplicity packing problems*. In such problems, it is not necessary to specify each rectangle individually in the input. Rather, for each distinct type of rectangle the input includes the dimensions of the type, as well as the number of rectangles of that type that must be packed. At first sight it might seem that these problem may have simpler solutions than instances with many rectangle types, however the input of such problems is very small and, therefore, it is more difficult to compute a solution in time bounded by a small function of the size of the input length.

All of the above packing problems can be generalized to higher dimensions. Rectangles, bins, and strips may have depths in addition to widths and heights, drastically increasing the difficulty of the problems.

1.2 Related Work

The bin packing and two-dimensional strip packing problems have been extensively researched. Since bin packing is equivalent to a strip packing problem where all items have equal height and bin packing is *NP*-hard [6], then strip packing is also *NP*-hard. For this reason, research on those problems has focused on approximation algorithms.

Let $A(I)$ be the value of the solution produced by an approximation algorithm A and let $OPT(I)$ be the value of an optimal solution for an instance I of some optimization problem. The approximation ratio of A is defined as the ratio $\frac{OPT(I)}{A(I)}$ if I is a maximization problem and $\frac{A(I)}{OPT(I)}$ if I is a minimization problem. For some problems there are approximation algorithms that produce solutions of value $A(I)$ satisfying $A(I) \leq C \times OPT(I) + o(OPT(I))$ for some value C, where the $o(OPT(I))$ term grows at a much slower rate than $OPT(I)$ as a function of the length of the input I. The value of C is known as the asymptotic approximation ratio of A. When $C = 1$ the approximation algorithm is known as asymptotically exact.

A simple algorithm for two-dimensional strip packing is the first-fit decreasing height (FFDH) algorithm. In this algorithm, rectangles are first sorted into order of non-increasing height; the first rectangles packed are the tallest and the last are the shortest. The first rectangle is placed at the bottom of the strip. This is the first "level" of the packing. Rectangles are packed on this level until the next rectangle does not fit. At this point, a new level is created on top of the tallest rectangle in the first level. Rectangles continue to be packed on the first level where they fit, creating new levels as necessary, until all rectangles are packed. First-fit is one of several algorithms in the class of "level-oriented" algorithms, which also includes next-fit (where previous levels are not revisited) and best-fit (where rectangles are packed on the level where they leave the least amount of empty space). FFDH was shown to have a solution quality at worst $\frac{17}{10}OPT(I) + 1$ by Coffman et al. [4] Over time, this bound has improved to $\frac{5}{3}OPT + \epsilon$ in Harren et al. [13], $\frac{4}{3}OPT + 7\frac{1}{18}$ in Golan [11], and $\frac{5}{4}OPT + \frac{53}{8}$ in Baker et al. [1]

Given an instance I of 2DSPP and a positive number ϵ, Kenyon and Rémila designed an algorithm [19] that produces a solution to I with height at most $(1 + \epsilon)OPT(I) + O(\frac{1}{\epsilon^2})$. The

time complexity of this algorithm is polynomial in both the number of rectangles and $\frac{1}{\epsilon}$. The algorithm is based on a linear programming relaxation of the problem that defines a fractional version of 2DSPP, where rectangles can be split into pieces with horizontal cuts. In Kenyon and Rémila's algorithm, rectangles are classified by width as being either "thin" or "wide" based on the value of ϵ. Wide rectangles are partitioned into groups and their widths are rounded to define a simpler fractional problem which can be solved efficiently. Once this fractional solution is obtained, any rectangles that are cut in the packing are simply "rounded up" such that they recover their original size. Thin rectangles are added back into the packing afterwards, utilizing the empty spaces left by the wide rectangles. Since 2DSPP is NP-hard, an asymptotic PTAS is the best one can do in polynomial-time assuming $P \neq NP$.

1.3 High-Multiplicity Packing Problems

In the high-multiplicity version of the strip packing problem (HMSPP), or *K-type problem*, there is a constant number K of rectangle types. Rectangles of the same type have the same width and height. Note that the input for a high multiplicity packing problem can be represented in a very compact manner. The input contains, for $1 \leq i \leq K$: w_i, the width of type i; h_i, the height of type i; and n_i, the number of items of type i. Hence, the input is specified by $3K$ numbers, regardless of how many rectangles need to be packed.

The high-multiplicity bin packing problem has been studied by Filippi and Agnetis [8] and they presented a polynomial-time, asymptotically exact algorithm. Given an input I, their algorithm produces outputs requiring at most $OPT(I) + K - 2$ bins. For the case where $K = 2$, an exact solution is obtained in time complexity $O(\log D)$ where D is the bin capacity. This problem is also known as the cutting stock problem, and has been researched extensively. A PTAS is known for this problem [19]. The best known approximation for the cutting stock problem for a constant number of object lengths is an $OPT + 1$ algorithm presented by Jansen and Solis-Oba [16].

Clifford and Posner in [2] investigated high-multiplicity parallel machine scheduling. In that paper, several polynomial time algorithms are presented for such machine scheduling problems. In addition, they show that NP-complete problems do not change complexity class when switching from a general encoding to a high-multiplicity encoding.

In Hoque [15], the rectangle packing problem is studied for the case where only two types of rectangles are packed. Hoque presents a PTAS for this problem and further claims the results can be generalized to any constant number of rectangle types.

To the best of our knowledge, there is no previously published work on the high-multiplicity strip packing problem (HMSPP).

1.4 Our Contributions

The 2DSPP is known to be NP-hard for an arbitrary number of distinct rectangle types. However, in practice there is often only a limited number of rectangle types. In this thesis we examine a version of the two-dimensional strip packing problem where the number of distinct rectangle types is constant.

The complexity of the high-multiplicity strip packing problem is unknown even for the case when the number of rectangle types is 2. We do not even know if HMSPP belongs to the class NP. Our main contribution is an $OPT(I) + K - 1$ polynomial-time approximation algorithm for the high-multiplicity strip packing problem. This improves on a simple algorithm for the problem which produces packings of height at most $OPT(I) + K$ in the worst case, but which was not known to run in polynomial time. We show here that such an algorithm does run in polynomial time. Although this may appear to be a small improvement, it is strong for small values of K; for example when $K = 2$ a solution of height at most $OPT(I) + 1$ is produced. As we show, moving from a solution of value $OPT(I) + K$ to one of value $OPT(I) + K - 1$ is not a trivial one, and it requires a significant amount of work. Furthermore, our bound almost matches the bound presented in Filippi and Agnetis [8] for high-multiplicity bin packing (i.e.,

strip packing when all rectangles have equal height), $OPT + K - 2$. If $OPT(I) \gg K$, then the difference in height between the approximation and the optimum is negligible, so our algorithm is asymptotically exact.

The reduction of 2DSPP to HMSPP does not make the problem easier to approximate in polynomial time. Because the problem can be represented using fewer bits, some techniques that are polynomial-time for 2DSPP are not polynomial-time for HMSPP; for example, we cannot make an individual decision for each rectangle.

Finally, although not all ideas in this thesis generalize to higher dimensions, several do. These may be a first step towards better approximations for the high-multiplicity strip packing problem of dimension $d \geq 3$.

In Chapter 2, we discuss our algorithm for HMSPP. This section includes a basic, high-level overview of the algorithm as well as an explanation of the fractional strip packing problem, linear programming, and a polynomial time algorithm for solving the fractional strip packing problem where the number of rectangle types is constant using a modified version of the Grötschel-Lovász-Schrijver algorithm. We also give a detailed description of our algorithm for two rectangles types and for any constant number of rectangles types. Chapter 3 consists of concluding thoughts, as well as open questions and future research.

Chapter 2

Algorithms

The most obvious way to solve HMSPP is, perhaps, to consider all possible packings for the rectangles and choose the best one. However, if there are many rectangles then the number of possible packings is very large and thus solving the problem using this method is slow and impractical. We wish to solve this problem with an algorithm with running time that is polynomial in the length of the input; i.e. an algorithm that is guaranteed to reach a solution and terminate in a number of steps that is a polynomial function of the length in bits of its input. As mentioned above, it is not known if HMSPP is in the complexity class P, or even in the class NP.

In this chapter we present a polynomial-time approximation algorithm for HMSPP that produces a solution of height at most $OPT(I) + K - 1$, where $OPT(I)$ is the height of an optimal solution and K is the number of rectangle types.

2.1 Algorithm Overview

Solving the fractional relaxation of HMSPP, where rectangles may be cut into pieces using horizontal cuts, is essential to our algorithm. This is typically solved using linear programming. Since as early as 1961 in Gilmore and Gomory [10], this has been a common strategy. The solution to the fractional problem is then converted into a solution to HMSPP. A solution

to the fractional problem consists of "configurations" and corresponding heights of each configuration to be packed. A configuration is a set of rectangles that fit together side-by-side in the strip. In a fractional solution, rectangles conforming to some configurations are packed upwards (as determined by solving a linear program) to certain heights where these rectangles are then cut, so that the next configuration used will begin packing at exactly that height; see Figure 2.1, where two configurations are packed.

In this thesis we use the word "configuration" to denote a set of rectangles of total width at most 1 and "packed configuration" to denote a section of the strip where rectangles are packed so that any horizontal line drawn across the section intersects the same multiset of rectangle types (i.e., a section of the packing where rectangles are packed according to some configuration).

There are several key ideas on which the main algorithm is built. First, rectangles can be rearranged horizontally within the same packed configuration (see Figure 2.1); this is a very useful property. When the number of distinct rectangle types is small relative to the total number of rectangles, the rectangles will often not need to be cut when switching between packed configurations in the fractional solution. If any two adjacent packed configurations contain rectangles of the same type, some of the rectangles in the upper packed configuration may be placed directly above rectangles of the same type in the lower packed configuration and this avoids the need to cut them when switching packed configurations. Instead of cutting these rectangles between the packed configurations, they can simply continue to be packed on top of one another until reaching a point where the type of rectangle must change (see Figure 2.1 (b)).

As mentioned before, in the fractional relaxation we are allowed to split rectangles into pieces using horizontal cuts. A simple way to convert a fractional solution of the strip packing problem into an integral one is to round up every cut rectangle to its original height. This will always result in trying to pack more rectangles than necessary. Therefore, in some parts of the fractional packing it is always possible to "round down" the cut rectangles and discard the fractional portion when the rest of the cut rectangles are "rounded up". This idea is discussed

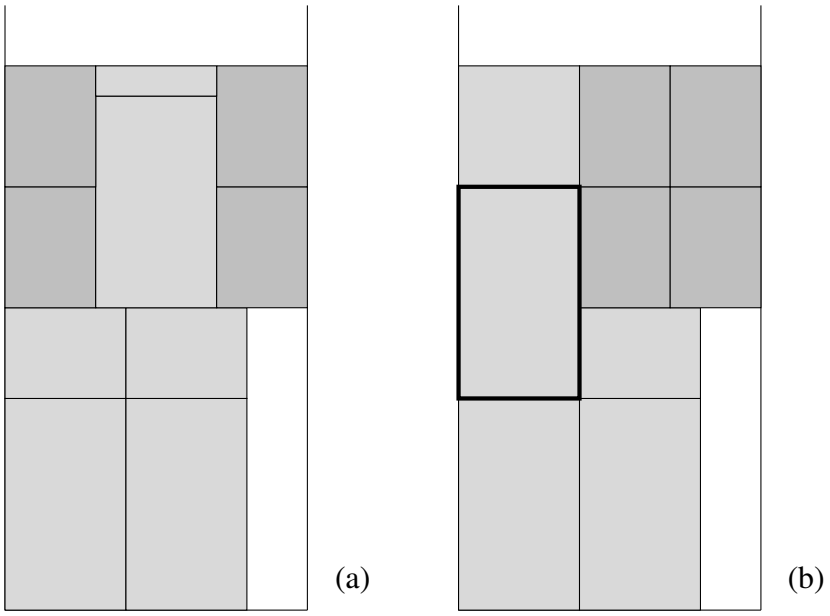

Figure 2.1: Two fractional solutions to the same problem. The horizontal ordering of the rectangles does not matter, both solutions have the same height. In Figure (b) the rectangle in bold does not need to be cut between packed configurations.

in detail in Sections 2.4 and 2.5.

Lastly, the change in height and width of the packing when rounding rectangles are inversely related. In fact, when combined with the previous assertion than we can "round down" the heights of rectangles in some circumstances, this allows us to bound the width of rounded rectangles as a function of the increase in height. Since the total area of the rectangles within the packing cannot change when converting fractional rectangles to whole rectangles, regions that contain rectangles that increase in height after becoming whole must become thinner (see Figure 2.2).

We use these ideas to show that any two packed configurations can be arranged such that, post-rounding, rectangles do not overlap and the height of the packing increases by at most one unit. To show how this is done we first consider the case when there are only two rectangle types and then we consider the more difficult case with a larger number of rectangle types.

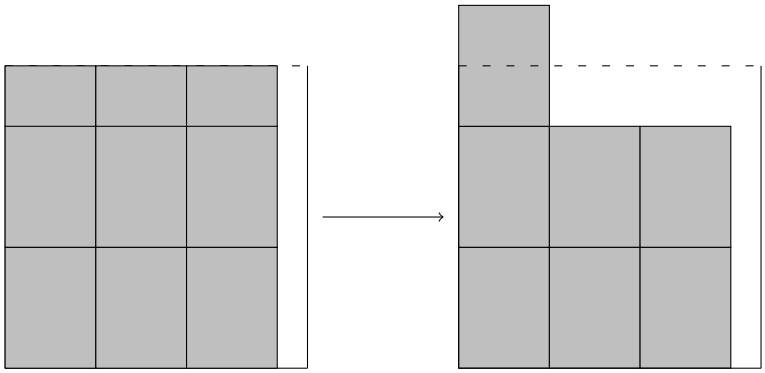

Figure 2.2: Before rounding, the area covered by rectangles intersecting the cutting line is equal to the area of 1.5 rectangles. If we round down this number, only one rectangle is packed. After rounding down, the total width of all rectangles intersecting the cutting line must decrease.

2.2 Fractional Strip Packing

Imagine that the requirements of HMSPP are relaxed so that rectangles may be sliced into multiple pieces with horizontal cuts. A solution to this "fractional" version of the two-dimensional strip packing problem can be expressed using a vector x of length equal to the total number of configurations. As indicated above, a configuration is a multiset of rectangles whose widths sum at most 1 and so they fit together side-by-side in the strip. Each element x_j of the vector is a rational number representing the height of the corresponding configuration within the solution; that is, each configuration C_j is packed to a height of x_j (see Figure 2.3).

Recall that n_i is the total number of rectangles of type T_i and h_i is the height of rectangles of type T_i. Let $\alpha_{i,j}$ be the number of rectangles of type T_i in configuration C_j, and let x_j denote the height to which configuration C_j should be packed. The fractional strip packing problem can be expressed as the following linear program:

$$\begin{aligned} \text{Minimize: } & \sum_j x_j \\ \text{Subject to: } & \sum_j x_j \alpha_{i,j} \geq n_i h_i, \text{ for each rectangle type } T_i \\ & x_j \geq 0 \end{aligned} \tag{2.1}$$

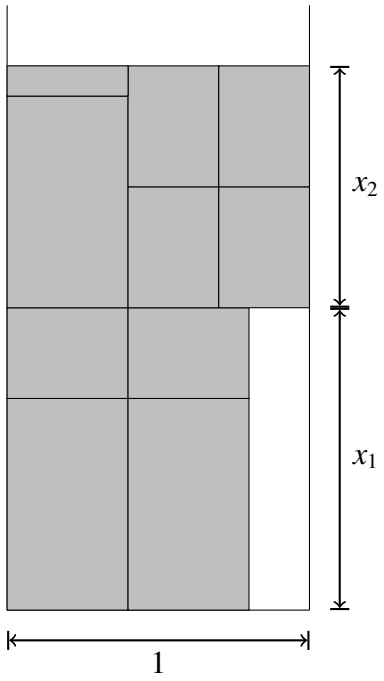

Figure 2.3: The fractional relaxation of the problem in Figure 1.1. Packed configuration C_1 consists of two rectangles of type T_1 and zero rectangles of type T_2 and packed configuration C_2 consists of one rectangle of type T_1 and two rectangles of type T_2.

In this section we present the first contribution of this paper, namely, a polynomial time algorithm for the fraction strip packing problem when the number of rectangle types is constant.

The set of feasible solutions, i.e. solutions that satisfy the constraints 2.1, can be thought of as a convex polytope of dimension equal to the number of variables in the linear program. A polytope is a geometric object with flat sides (e.g., a polygon is a two-dimensional polytope). This polytope is known as the *feasible region*. Each face of the polytope corresponds to a constraint. The problem of solving a linear program is equivalent to finding a point within the feasible region that optimizes the objective function. The vertices of the polytope are known as basic feasible solutions. If a linear program has an optimal solution, then it has at least one basic feasible solution that optimizes the objective function. It is known that in any basic feasible solution, the number of nonzero variables is at most the lesser of the number of constraints (excluding nonnegativity constraints) and the number of variables [17]. Thus, the number of nonzero variables in a basic feasible solution to (2.1) is no greater than K, the

number of constraints. Therefore, the number of configurations used in a basic feasible solution for (2.1) is at most the number of rectangle types, K.

Recall that $OPT(I)$ is the height of an optimal packing for instance I of HMSPP. Let $LIN(I)$ denote an optimal solution to the fractional strip packing associated with I and $AREA(I) = \sum_r w_r h_r n_r$ be the total area of all rectangles in I. Then is is not hard to see that

$$AREA(I) \leq LIN(I) \leq OPT(I).$$

The number of variables in linear program (2.1) is equal to the number of configurations, which is at most n^K, and the number of constraints is determined by the number of rectangle types. Solving the linear program will yield an optimal solution for the fractional strip packing problem. However, solving a linear program with many variables can be computationally too expensive.

2.3 The GLS Algorithm

We wish to solve linear program (2.1) when the number of rectangle types is constant. However, it is not good enough to solve it in time polynomial in the number of rectangles because this may be exponential in the number of bits needed to encode an instance of the problem. In 2DSPP, each rectangle's width and height are given individually in the input. However, in HMSPP the input is much more compact; the problem contains only K widths, heights, and numbers of rectangles.

The fractional strip packing problem is identical to the fractional bin packing problem; in the latter problem a configuration is a set of items that fit together within a single bin and the solution to the linear program will give us the fractional number of bins in which each configuration should be packed. Therefore, we can borrow several techniques from Karmarkar and Karp [18] to solve (2.1) in polynomial time.

The linear program (2.1) has a constant number of constraints and one variable for ev-

ery configuration. However, as mentioned above the number of configurations, and therefore variables, is potentially very large which makes solving the problem difficult. Therefore, we consider the dual linear program instead. Recall that every linear program has a dual linear program and both have optimal solutions of the same value [17]. The dual linear program of 2.1 has a number of constraints $O(n^K)$, but only a constant number K of variables.

The dual program of (2.1) can be interpreted as having variables v_j representing the values of some items j. The objective of this linear program is to maximize the total value of all items and it is constrained such that values must be nonnegative and the total value of items within any configuration must be at most 1. The dual program of (2.1) is the following:

$$\begin{aligned} &\text{Maximize: } \sum_i v_i n_i \\ &\text{Subject to: } \sum_i v_i \alpha_{i,j} \leq 1, \text{ for each configuration } C_j \\ &\qquad v_i \geq 0 \end{aligned} \tag{2.2}$$

Linear program (2.1) has K constraints and $O(n^K)$ variables. The dual (2.2) has K variables and $O(n^K)$ constraints and it can be written in the form

$$\begin{aligned} &\max\ c^T v \\ &\text{s.t. } v^T A \leq 1 \\ &\qquad v \geq 0 \end{aligned} \tag{2.3}$$

We use the Grötschel-Lovász-Schrijver (GLS) algorithm [12], which is an iterative algorithm based on the ellipsoid method, to solve the dual program (2.3). This algorithm finds a solution for (2.3) of value $LIN(I) + h$, where h is an additive tolerance that affects the running time of the algorithm. One important element of the algorithm is a "separation oracle" which given a potential solution z within the current iteration of the GLS algorithm determines whether or not the solution is feasible and, if it is not, returns a constraint that is violated by

z. For fractional strip packing, this oracle needs to find an optimal solution for the knapsack problem.

2.3.1 The Separation Oracle

Recall that an instance of the knapsack problem consists of items with weights and values. The goal is to pack items into a knapsack of fixed capacity such that items in the knapsack do not exceed the capacity and their total value is maximized. Let s_j be the weight of item type j, v_j be the value of item type j, n_j be the number of items of type j, and S be the capacity of the knapsack. The knapsack problem can be expressed as the following integer program where x_j is the number of items of type j to be placed in the knapsack:

$$\begin{aligned} \text{Maximize: } & \sum_j v_j x_j \\ \text{Subject to: } & \sum_j s_j x_j \leq S \\ & x_j \in \left\{0, 1, ..., n_j\right\} \end{aligned} \tag{2.4}$$

Lenstra [20] proved that integer programming with a fixed number of variables can be solved in polynomial time. An integer program defined by m constraints involving numbers of at most b bits can be solved using Eisenbrand's algorithm [7] in expected time $O(m + b \log m)$ when the number of variables is fixed and in time $O(b)$ when the number of constraints is also fixed. Not only do knapsack problems have only one constraint, but for knapsack problems arising from the GLS algorithm for HMSPP they also have at most a constant number K of variables. Hence, we can solve these instances of the knapsack problem in time $O(b)$.

2.3.2 Solving the Dual Linear Program

If $n \geq 2$ and $K \geq 2$, then $n^K > K$. Hence, since the number of variables in (2.3) is K, the largest set of linearly independent rows of A has K rows. Let B be a $K \times K$ matrix formed by K linearly independent rows of A. Then a solution to the linear program

$$\begin{aligned} &\max c^T v \\ &\text{s.t. } v^T B \leq 1 \\ &v \geq 0 \end{aligned} \tag{2.5}$$

is also a basic feasible solution (BFS) for (2.3) and in fact every basic feasible solution for (2.3) is a solution for a linear program of the form (2.5).

By Lemma 2.1 of [22], if $x = (x_1, x_2, \ldots, x_K)$ is a BFS for (2.5) then each x_i is a rational number $\frac{N_i}{D_i}$ where the absolute value of N_i and D_i is bounded by $L = K!\beta^{K-1}$, where β is the maximum value of any entry of B. Since $\beta \leq n$, then $L \leq K!\, n^{K-1} < (nK)^{K-1}$ and $b \leq \log\left((nK)^{K-1}\right)$.

This means that if we choose the additive tolerance to be $h = (nK)^{-K+1}$ then a solution for (2.5) of value at most $LIN(I) + h$ computed by the GLS algorithm must in fact have value $LIN(I)$, where $LIN(I)$ is the value of an optimal solution for (2.5). Therefore, a basic feasible solution for (2.3) computed by the GLS algorithm with tolerance $(nK)^{-K+1}$ must be an optimal basic feasible solution for (2.3).

We can transform a solution for linear program (2.3) into a basic feasible solution for (2.1) using the algorithm of Karmarkar and Karp with tolerance $h = (nK)^{-K+1}$. Let a be the width of the thinnest rectangle. The complexity of Karmarkar and Karp's algorithm is

$$O\left(K^8 \ln K \log^2\left(\frac{(Kn)^K}{a}\right) + K^2 \log K(K \log n + K \log K)\right)$$

$$= O\left(K^{10} \log K \log^2\left(\frac{Kn}{a}\right) + K^3 \log K \log n\right).$$

For the rest of the paper we assume that a basic feasible solution is known for (2.1) and it consists of at most K configurations stacked one on top of the other.

A naive algorithm for the high multiplicity strip packing problem would compute a basic feasible solution for (2.1) and then simply round each fractional packed configuration up such that any cut rectangles recover their full height. Based on the results of this section, such an algorithm clearly runs in polynomial time and it computes a solution to HMSPP of height at most $OPT(I) + K$.

2.4 The Two-Type Strip Packing Problem

In this section we consider the high-multiplicity strip packing problem for $K = 2$. A basic feasible solution to the linear program (2.1) for an instance of the two-type strip packing problem has a particular structure, as that (fractional) solution packs all rectangles using only two configurations C_1, C_2 as shown in Figure 2.4. These packed configurations can be divided into three distinct sections horizontally. The outer two sections consist of rectangles of the same type and, hence, they do not need to be cut when switching between configurations. Rectangles in this part of the solution can be packed on top of each other until reaching the top of the packing. The middle section occurs where there are two different types of rectangles and so the rectangles in the lower packed configuration might need to be cut at the border of the packed configurations. All rectangles in the upper packed configuration also might be cut at the top of the packing.

The rectangles in the outer sections that appear at the top of the packing are simply rounded up such that they are whole. This rounding increases the height of these outer sections by at most 1 unit. In the middle section, the type of rectangles in the lower packed configuration is not the same as the type of rectangles in the upper packed configuration. We cannot simply round up rectangles at the borders of both packed configurations without increasing the height of the packing by up to 2 units. Let M_j be the set of rectangles in the middle section that

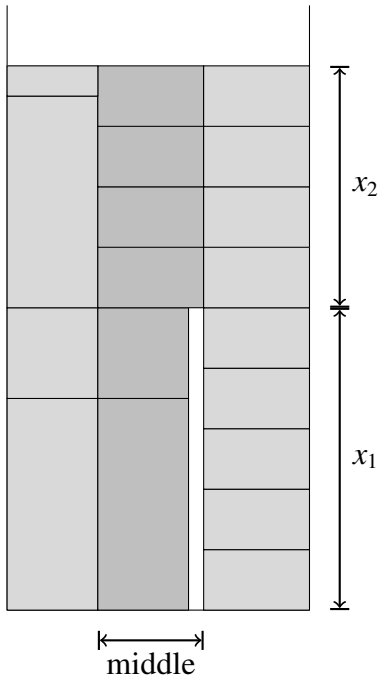

Figure 2.4: A packing of rectangles of 2 types. Note that there are three distinct sections horizontally: outer sections which each consist exclusively of rectangles of one type, and a middle section where the lower packed configuration has rectangles of a different type than the upper packed configuration.

intersect the cutting line at the top of packed configuration C_j for $j = 1, 2$. Assume without loss of generality that the rectangles in the middle section of the lower packed configuration, C_1, are of type T_1 and the rectangles in the middle section of the upper packed configuration, C_2, are of type T_2.

The rectangles within the middle section that are cut in the fractional solution represent some number, not necessarily integer, of rectangles; e.g. three rectangles cut at half their height represent 1.5 rectangles of that type. Let this amount be n'_j for rectangles in M_j. Obviously, we cannot have fractional rectangles in the final solution to an HMSPP instance. Therefore, we either round up or round down n'_j to an integer value and pack that many whole rectangles of type T_j instead of the fractional rectangles in M_j. In particular, if we round up rectangles in M_j we pack $\lceil n'_j \rceil$ whole rectangles of type T_j and if we round them down we pack $\lfloor n'_j \rfloor$.

Lemma 2.4.1 *If the number of rectangles in the outer sections are rounded up, then n'_j can be*

rounded down for both types T_j.

Proof Let n'_j be the number of rectangles represented by the cut rectangles in M_j and o'_j be the number of rectangles represented by the cut rectangles of type T_j in the rest of the packing. We know $n'_j + o'_j \geq z$, where z is a integer representing the number of whole rectangles needed to replace all fractional ones. Therefore, $\lfloor n'_j \rfloor + \lceil o'_j \rceil \geq z$. □

Rounding down the middle section of both packed configurations allows us to pack the rounded rectangles into a smaller space than if we were to round either or both up. This is essential to the algorithm.

Let $f_j h_j$ be the height of the fractional rectangles of type T_j that are cut and lie below the cutting line located at the top of packed configuration C_j. Imagine that the rectangles in M_1 were removed from the packing. This would reduce the height of the middle section by $f_1 h_1$. Adding these fractional rectangles to the top of the packing (still in the middle section) would then increase the height of the packing by the same amount. The total height of the section does not change, the fractional rectangles are simply moved. We show that these fractional rectangles can be made whole by rounding their heights in such a way that they do not increase the height of the middle section by more than 1 unit.

We consider 2 cases:

Case 1. $f_1 + f_2 > 1$. We round up the heights of the rectangles in M_j to h_j for both types. This increases the height of the packing in the middle section by at most

$$(1 - f_1)h_1 + (1 - f_2)h_2 \leq 1 - f_1 + 1 - f_2 = 2 - (f_1 + f_2) < 1.$$

Case 2. $f_1 + f_2 \leq 1$. Since the area of the fractional rectangles in M_j is at most $f_j h_j W$, where W is the width of the middle section and the number n'_j was rounded down, the total area of the rectangles of type T_j in M_j cannot be more than $f_j h_j W$. If the height of these fractional rectangles is rounded up to h_j their total width would be at most $f_j W$. If we put the rounded rectangles in M_1 and M_2 side by side, their total width is at most $f_1 W + f_2 W = (f_1 + f_2)W \leq W$

so they fit in the middle section and this packing of the rounded rectangles increases the height of the middle section by at most $max\{h_1, h_2\} \leq 1$.

Summarizing, after finding an optimal fractional packing, we round up the outer sections, move rectangles cut in the middle section of C_1 to the top of the middle section of C_2, and round down rectangles cut in the middle section as described above. Each of these steps can be performed in constant time. Therefore, the worst-case runtime of this algorithm is dominated by that of solving the fractional strip packing problem, which is

$$O\left(K^{10} \log K \log^2\left(\frac{Kn}{a}\right) + K^3 \log K \log n\right),$$

as described in the previous section.

Theorem 2.4.2 *Let $OPT(I)$ be the height of an optimal solution for an instance I of the two-type strip packing problem. There is a polynomial-time algorithm A which, given I, produces a packing of I in a strip of width* 1 *and height $A(I)$ such that:*

$$A(I) \leq OPT(I) + 1.$$

2.5 The K-Type Strip Packing Problem

A basic feasible solution for linear program (2.1) for K types of rectangles uses at most K configurations. As mentioned above, simply rounding each packed configuration up in the fractional solution such that all rectangles are integral would increase the height of the packing by at most K. However, this is not the best one can do. In this section we present an algorithm for the K-type strip packing problem that for any instance I produces a packing for I of height at most $OPT(I) + K - 1$.

Let a basic feasible solution for (2.1) with K types use configurations $C_1, C_2, \ldots, C_K$. Let us consider a packing S where rectangles are packed according to configurations C_i and these configurations are stacked one on top of the other in the strip.

Lemma 2.5.1 *Any two adjacent packed configurations C_i, C_{i+1} can have their rectangles rearranged horizontally such that there is a contiguous section of the packed configurations where rectangles of the same type line up between the packed configurations and hence they are not cut at the boundary between C_i and C_{i+1}. In the remaining section of the packed configurations, no type of rectangle will appear in more than one packed configuration.*

Proof For each type T_r, let m_r be the minimum of the number of rectangles of type T_r across C_i (i.e., the number of rectangles packed side by side in C_i) and the number of rectangles of type T_r across C_{i+1}. In the left-most portion of each packed configuration, which we will call section S_1, m_r rectangles are packed across for each type T_r. The rest of the rectangles are packed to the right of S_1, in what we call section S_2 (see Figure 2.5). Let $t_{r,q}$ be the total number of rectangles of type T_r across packed configuration C_q for $q = i, i+1$. The number of rectangles of type T_r packed across section S_2 of packed configuration C_q is $t_{r,q} - m_r$. Note that for all r, in at least one of the packed configurations C_q the value of $t_{r,q}$ will be zero. Therefore, no type T_r will appear in section S_2 of both packed configurations. □

Each type of rectangle may be cut several times in a fractional packing. Let $a_{r,q}$ be the area of the rectangles of type T_r in packed configuration C_q that are cut. Recall that w_r and h_r are the width and height of rectangle type T_r, respectively. Then, $n_{r,q} = \frac{a_{r,q}}{w_r h_r}$ is the (potentially fractional) number of rectangles of type T_r that must be packed to cover the area of the rectangles sliced by the cutting line in packed configuration C_q. Even though this number may be fractional, the total number of these rectangles in all cuts, $n_r = \sum_{i=1}^{K} n_{r,i}$, is integral for any type T_r. When rectangles cut in packed configuration C_q are rounded up, the cut rectangles are replaced with $\lceil n_{r,q} \rceil$ whole rectangles. This will increase the height of the packed configuration because whole rectangles will be taller than those which are cut.

Lemma 2.5.2 *Define a segment as a set of rectangles of the same type that intersect the cutting line within a section S_2 of a packed configuration C_q. For all types T_r, at least one segment formed by rectangles of type T_r can be rounded down such that $\lfloor n_{r,q} \rfloor$ rectangles are packed for*

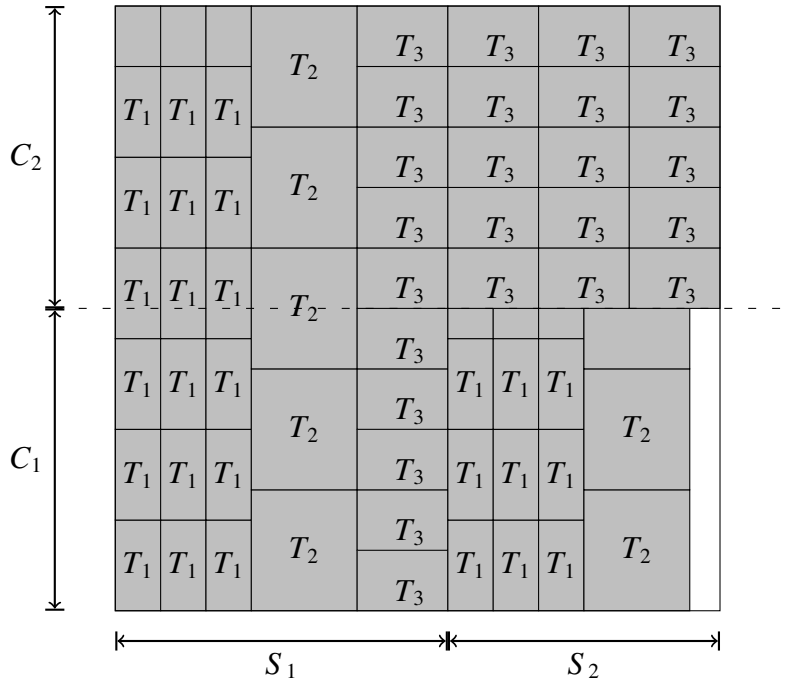

Figure 2.5: An example of an arrangement of three rectangle types in two packed configurations. Rectangles that appear in both are placed to the left in section S_1. In the remaining section, S_2, no type appears in both packed configurations.

that segment while the number of rectangles of type T_r in the rest of the packing are rounded up. The total number of rectangles of type T_r in the entire final packing will be n_r.

Proof Let $n'_{r,q}$ be the total (fractional) number of rectangles of type T_r cut at all cutting lines except those cut at the top of section S_2 of some packed configuration C_q. Rounding up the rectangles of type T_r in all packed configurations except those in section S_2 of C_q will pack at least $\lceil n'_{r,c} \rceil$ rectangles. If we round down the rectangles of type T_r cut in section S_2 of C_q, then $\lfloor n_{r,q} \rfloor$ rectangles are packed in their place. Because $n'_{r,q} + n_{r,q}$ is integer, the amount gained from rounding rectangles up will always be at least the decrease from rounding down $n_{r,q}$ and therefore $n'_{r,q} + n_{r,q} \leq \lceil n'_{r,q} \rceil + \lfloor n_{r,q} \rfloor = n_r$, so the total number of whole rectangles of type T_r that are packed is n_r. □

Lemma 2.5.2 is a generalization of Lemma 2.4.1.

Consider two adjacent packed configurations C_1 and C_2 where C_2 is directly on top of C_1 in packing S. All other packed configurations in S are rounded up such that cut rectangles are

made whole and each increases in height by no more than 1 unit, increasing the total height of the packing by no more than $K-2$. The rectangles in the two chosen packed configurations are arranged as described in Lemma 2.5.1 into sections S_1 and S_2. Rectangles in section S_1 need not be cut between C_1 and C_2 and rounding needs to be done only at the top of C_2. Additionally, in the section S_2 where rectangles do not line up between packed configurations, let $f_{r,q}$ be the fraction of each rectangle cut of type T_r that lies below the cutting line in packed configuration C_q for $q = 1, 2$ (so the pieces of rectangles of type T_r that were cut and appear at the top of packed configuration C_q all have height $f_{r,q}h_r$). Let S_2' be the part of S_2 where these fractional rectangles are packed.

The rectangles in section S_2 of packed configurations C_1 and C_2 are sorted within that section by nondecreasing values $f_{r,1}$ and $f_{r,2}$, respectively. Following Lemma 2.5.2, the number of fractional rectangles in S_2' may be rounded down since no type appears in both C_1 and C_2 of section S_2. After this rounding we replace the fractional rectangles with whole ones. After doing this, in some places within section S_2 the height of the packing may exceed the pre-rounding fractional height and in some other places it may have decreased (see Figure 2.6). The portions of the packed configurations that have not increased in height are placed to the right of section S_2 (see Figure 2.6).

Let $\omega_{r,q}$ be the total width of all rectangles of type T_r in section S_2' of packed configuration C_q and let $\omega'_{r,q}$ be the total width of these rectangles after the rounding. Because the whole rectangles packed in S_2' after performing the rounding have total area less than or equal to the total area occupied by the fractional rectangles, the total width of these whole rectangles is at most $f_{r,q}$ times the total width of the fractional rectangles and, therefore, $\omega'_{r,q} \leq f_{r,q}\omega_{r,q}$. Furthermore, replacing fractional rectangles with whole ones will increase the height of the packing by the difference between their fractional height and the height of rectangles of that type: $h_r - f_{r,q}h_r \leq (1 - f_{r,q})h_r$.

We argue that for both packed configurations C_1 and C_2, all whole rectangles that replace the fractional ones after the rounding will fit within the original packed configurations plus

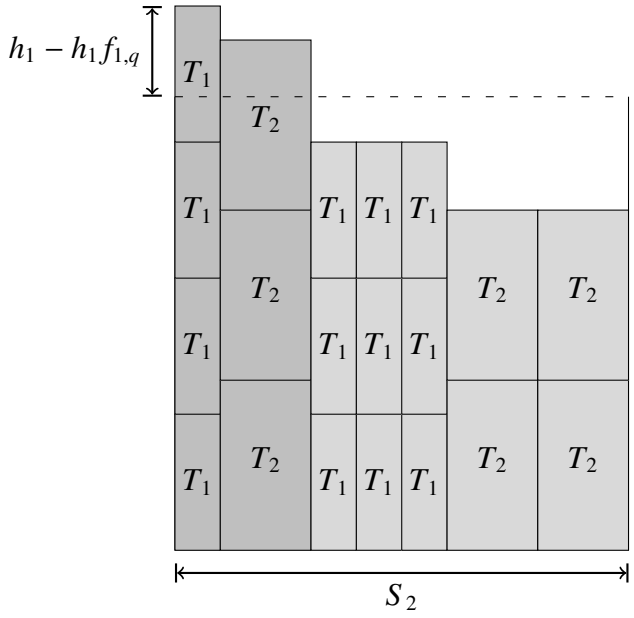

Figure 2.6: An example of section S_2 of a packed configuration C_q after "rounding down". Because $f_{1,q} \leq f_{2,q}$, type T_1 is packed first. The dotted line represents the original fractional height of the packed configuration. Portions that do not increase in height are packed in the far right of the section so that portions of the packed configuration that increase in height are directly adjacent to one another.

triangular regions directly on top of each packed configuration. These regions are right triangles with bases of width W_2 equal to the width of section S_2 and heights of 1 unit formed by a line that intersects the upper-right corner of a packed configuration and a point one unit above the left side of section S_2 of the packed configuration (see Figure 2.7). Let us consider packed configuration C_1. Let $T_1, T_2, \ldots, T_{C_1}$ be the types of the rectangles in section S_2 of C_1 in nondecreasing order of $f_{r,1}$ value.

Assuming a coordinate system as shown in Figure 2.7, the equation of the line defining the hypotenuse of the triangular region is $y = 1 - \frac{1}{W_2}x$. If the upper-right corner of a rectangle r_i fits within the triangular region, then the entire rectangle fits. For a type T_r, this point has an x coordinate x_i equal to w_r plus the sum of the widths of all rectangles preceding it within section S_2 and a y coordinate y_i equal to its height, h_r, minus $f_{r,q}h_r$. Thus, rectangle r_i is within the triangular region if and only if $y_i + \frac{1}{W_2}x_i \leq 1$. The rounded rectangles of type T_r fit in the

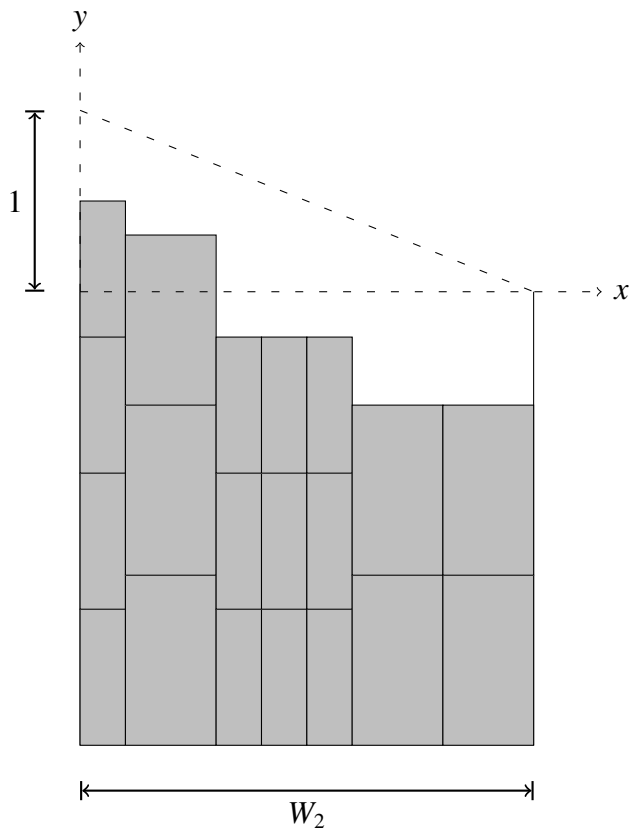

Figure 2.7: All rectangles will fit in the depicted region after rounding.

triangular region above packed configuration C_1 if and only if:

$$h_r(1 - f_{r,1}) + \frac{1}{W_2} \sum_{i=1}^{r-1} \omega'_{i,1} \leq 1.$$

Since the height of any rectangle is at most 1, the above condition holds if

$$1 - f_{r,1} + \frac{1}{W_2} \sum_{i=1}^{r-1} \omega'_{i,1} \leq 1.$$

Since $\omega'_{r,1} \leq f_{r,1}\omega_{r,1}$, the inequality is satisfied if

$$1 - f_{r,1} + \frac{1}{W_2} \sum_{i=1}^{r-1} f_{i,1}\omega_{i,1} \leq 1.$$

Because rectangle types are placed in nondecreasing order of $f_{r,1}$ and if $i \leq r-1$, then $f_{i,1} \leq f_{r,1}$,

the inequality is true if

$$1 - f_{r,1} + \frac{1}{W_2} \sum_{i=1}^{r-1} f_{r,1}\omega_{i,1} \leq 1.$$

Or equivalently,

$$1 - f_{r,1} + \frac{f_{r,1}}{W_2} \sum_{i=1}^{r-1} \omega_{i,1} \leq 1.$$

The total width of rectangles within section S_2 cannot exceed the width of section S_2, so $\sum_{i=1}^{r-1} \omega_{i,1} \leq W_2$. Hence, the above inequality holds true because

$$1 - f_{r,1} + f_{r,1} \leq 1.$$

Therefore, rectangles of all types will fit within the region when packed in this manner. The same argument can be applied to C_2.

Post-rounding, if one packed configuration is arranged upside down (as if it were turned 180°) and placed above the other packed configuration, then the triangular regions will not overlap and the total increase in height will be at most 1 (see Figure 2.8). Section S_1 increases in height by a most 1 unit and S_2 increases by at most 1, so packed configurations C_1 and C_2 together increase the height of the packing by at most 1 unit. The rest of the packed configurations were rounded up and together they increase the height of the packing by at most $K - 2$. Therefore, the total height of the final packing is at most $K - 2 + 1 = K - 1$.

Theorem 2.5.3 *Let $OPT(I)$ be the height of an optimal solution for an instance I of the K-type strip packing problem. There is a polynomial-time algorithm A which, given I, produces a packing of I in a strip of width* 1 *and height $A(I)$ such that:*

$$A(I) \leq OPT(I) + K - 1.$$

Proof The algorithm described above computes a packing of the desired height, so we only need to show that it runs in polynomial time. The first step of the algorithm is to compute a basic feasible solution for (2.1) and that can be done using the algorithm described in Section 2.3.

Recall that a is the width of the thinnest rectangle; the time complexity of this algorithm is dominated by the solution of the linear program, which requires

$$O\left(K^{10}\log K\log^2\left(\frac{Kn}{a}\right)+K^3\log K\log n\right)$$

time in the worst case. Once we have solved the linear program, we choose two packed configurations and arrange their rectangles into sections S_1 and S_2 as described in the proof of Lemma 2.5.1. This is done in constant time. Items in section S_2 are sorted, and because we are sorting rational numbers, we can use radix sort which runs in $O(cN)$ time where c is the average number of bits per item and N is the number of items to be sorted. In our algorithm, $N \leq K$ which is constant. By Lemma 2.1 of [22] (see also Section 2.3.2), $c \leq K\log n + K\log K$. After items are sorted, some rearrangement takes place, all in constant time. Therefore, the total runtime of the algorithm in the worst case is

$$O\left(K^{10}\log K\log^2\left(\frac{Kn}{a}\right)+K^3\log K\log n\right).$$

□

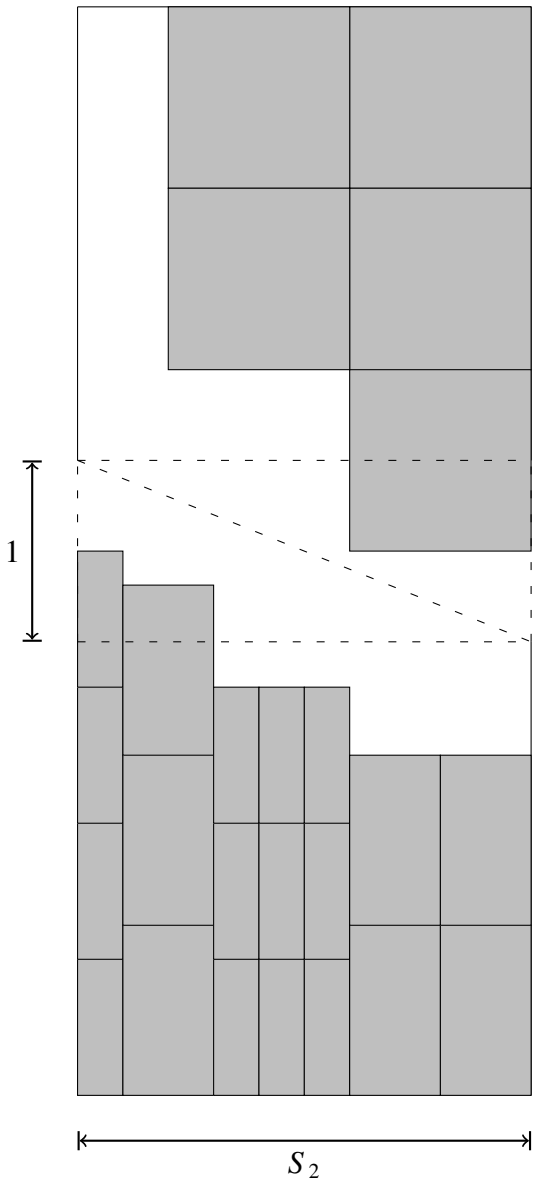

Figure 2.8: After rounding, section S_2 of packed configuration C_2 is turned upside down and placed above section S_2 of packed configuration C_1. Since the triangular regions fit together nicely, the total increase in height is at most 1 unit.

Chapter 3

Conclusion

In this thesis we considered the high-multiplicity strip packing problem. In this problem, we are given an input I consisting of the width, height, and number of rectangles for each of K rectangle types. HMSPP and related packing problems have many applications in diverse fields including transportation, stock cutting, printing, and scheduling. We present an approximation algorithm for HMSPP that produces a solution of height at most $OPT(I) + K - 1$ in time $O\left(K^{10} \log K \log^2\left(\frac{Kn}{a}\right) + K^3 \log K \log n\right)$, where $OPT(I)$ is the value of an optimal packing for I, K is the number of rectangle types, n is the total number of rectangles, and a is the width of the thinnest rectangle.

3.1 Open Questions and Future Work

Our results give way to many new questions. The complexity class for HMSPP is still not known even for small values of K. It is not even known if HMSPP is in the class NP. Assuming the problem is not in class P, is $OPT(I) + K - 1$ the best one can do in polynomial time? The high-multiplicity bin packing problem is in class P when $K = 2$ [21]. However, it is not known if HMSPP can be solved exactly in polynomial time for any small values of $K \geq 2$.

High-multiplicity strip packing problems are not strictly limited to two dimensions. Our algorithm is not trivially generalized to higher dimensions, however some of the key ideas

could be used. Particularly, three-dimensional fractional strip packing problems are not easily represented as linear programs, a very necessary part of our algorithm.

In our algorithm, only two packed configurations are examined and rearranged after solving the fractional problem. This is because Lemma 2.5.2 does not allow us to round down more packed configurations. If the packed configurations were divided into pairs, each pair could be arranged as described above such that each pair increased in height by only 1 unit assuming some fractional rectangles were removed from the packed configurations and placed aside. If there were a way to pack these leftover rectangles in a height less than $\frac{K}{2}$ then the resulting algorithm would improve on the $OPT(I) + K - 1$ algorithm we have presented in this thesis. However, so far we have not found a way to pack the leftover rectangles in such a space.

Bibliography

[1] Brenda S. Baker, Donna J. Brown, and Howard P. Katseff. A $\frac{5}{4}$ algorithm for two-dimensional packing. *Journal of Algorithms*, 2(4):348–368, 1981.

[2] John J. Clifford and Marc E. Posner. Parallel machine scheduling with high multiplicity. *Mathematical programming*, 89(3):359–383, 2001.

[3] Edward G. Coffman, Jr, Michael R. Garey, and David S. Johnson. An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, 7(1):1–17, 1978.

[4] Edward G. Coffman, Jr., Michael R. Garey, David S. Johnson, and Robert Endre Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980.

[5] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[6] W. Fernandez De La Vega and George S. Lueker. Bin packing can be solved within $1+\varepsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

[7] Friedrich Eisenbrand. *Fast integer programming in fixed dimension*. Springer, 2003.

[8] Carlo Filippi and Alessandro Agnetis. An asymptotically exact algorithm for the high-multiplicity bin packing problem. *Mathematical programming*, 104(1):21–37, 2005.

[9] Aleksei V. Fishkin, Olga Gerber, Klaus Jansen, and Roberto Solis-Oba. Packing weighted rectangles into a square. In Joanna Jdrzejowicz and Andrzej Szepietowski, editors, *Mathematical Foundations of Computer Science 2005*, volume 3618 of *Lecture Notes in Computer Science*, pages 352–363. Springer Berlin Heidelberg, 2005.

[10] Paul C. Gilmore and Ralph E. Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

[11] Igal Golan. Performance bounds for orthogonal oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 10(3):571–582, 1981.

[12] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[13] Rolf Harren, Klaus Jansen, Lars Prädel, and Rob Van Stee. A $(5/3+\varepsilon)$-approximation for strip packing. In *Algorithms and Data Structures*, pages 475–487. Springer, 2011.

[14] Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)*, 34(1):144–162, 1987.

[15] Sheik Mamun-Ul Hoque. The 2-rectangle packing problem. Master's thesis, The University of Western Ontario, 2011.

[16] Klaus Jansen and Roberto Solis-Oba. An $OPT+1$ algorithm for the cutting stock problem with constant number of object lengths. In *Integer Programming and Combinatorial Optimization*, pages 438–449. Springer, 2010.

[17] Howard Karloff. *Linear programming*. Springer, 2008.

[18] Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 312–320. IEEE, 1982.

[19] Claire Kenyon and Eric Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25(4):645–656, 2000.

[20] Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, pages 538–548, 1983.

[21] S. Thomas McCormick, Scott R. Smallwood, and Frits CR Spieksma. A polynomial algorithm for multiprocessor scheduling with two job lengths. *Mathematics of Operations Research*, 26(1):31–49, 2001.

[22] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications, 1998.

Curriculum Vitae

Name:	Devin Price
Post-Secondary Education and Degrees:	Gardner-Webb University North Carolina 2010 - 2012 B.Sc. University of Western Ontario London, ON 2012 - 2014 M.Sc.
Related Work Experience:	Teaching Assistant The University of Western Ontario 2012 - 2013