

# Chips on Wafers, or Packing Rectangles into Grids

Mattias Andersson<sup>b</sup> Joachim Gudmundsson<sup>a,1</sup>  
Christos Levcopoulos<sup>b</sup>

<sup>a</sup>*Department of Mathematics and Computing Science, TU Eindhoven, 5600 MB Eindhoven, The Netherlands. Email: h.j.gudmundsson@tue.nl*

<sup>b</sup>*Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden. Email: mattias@cs.lth.se, christos@cs.lth.se*

---

## Abstract

A set of rectangles  $\mathcal{S}$  is said to be *grid packed* if there exists a rectangular grid (not necessarily regular) such that every rectangle lies in the grid and there is at most one rectangle of  $\mathcal{S}$  in each cell. The area of a grid packing is the area of a minimal bounding box that contains all the rectangles in the grid packing. We present an approximation algorithm that given a set  $\mathcal{S}$  of rectangles and a real constant  $\varepsilon > 0$  produces a grid packing of  $\mathcal{S}$  whose area is at most  $(1 + \varepsilon)$  times larger than an optimal grid packing in polynomial time. If  $\varepsilon$  is chosen large enough the running time of the algorithm will be linear. We also study several interesting variants, for example the smallest area grid packing containing at least  $k \leq n$  rectangles, and given a region  $\mathcal{A}$  grid pack as many rectangles as possible within  $\mathcal{A}$ . Apart from the approximation algorithms we present several hardness results.

---

## 1 Introduction

In the VLSI wafer industry it is nowadays possible that multiple projects share a single fabrication matrix (the wafer); this permits fabrication costs to be shared among the participants. No a priori constraints are placed on either the size of the chips nor on the aspect ratio of their side lengths (except the maximum size of the outer bounding box). After fabrication, in order to free the separate chips for delivery to each participant, they must be cut from the wafer. A diamond saw slices the wafer into single chips. However cuts can only be made all the way across the bounding box, i.e., all chips

---

<sup>1</sup> Supported by The Netherlands Organisation for Scientific Research (NWO).

must be placed within a grid. A grid is a pattern of horizontal and vertical lines (not necessarily evenly spaced) forming rectangles in the plane. There are some practical constraints, for example, the distance between two parallel cuts cannot be infinitely small, since machines with a finite resolution must be programmed with each cut pattern. Although some of these constraints may simplify the problem we will not consider them in this paper.

We will consider the theoretical aspects of the problem and we simplify it by assuming that the wafers are rectangular. In practice the wafers are usually circular, although rectangular wafers are produced and used in small scale [15,19]. This application leads us to define grid packing as follows.

**Definition 1** *A set of rectangles  $\mathcal{S}$  is said to be grid packed if there exists a rectangular grid such that every rectangle lies in the grid and there is at most one rectangle of  $\mathcal{S}$  in each cell, as illustrated in Fig. 1. The grid is said to enclose  $\mathcal{S}$  if it contains no empty rows or columns and the height/width of every row/column is minimized with respect to its included rectangles. The area of a grid packing is the area of a minimal bounding box that contains all the rectangles in the grid packing.*

There are several cases in the chip design industry where it is favorable to produce a good grid packing. Some practitioners have therefore asked for algorithms solving such problems [12]. In practice one is given a set of chips and one wants to minimize the number of wafers used. A naïve simulated-annealing rectangle placement [12] with  $n$  rectangles might be packed on  $n$  separate wafers, as a worst case. This means, fabricating  $n$  wafers, wasting all except one chip on each wafer. So the number of recovered chips per cut pattern should be maximized.

Note that the number of input rectangles is  $n$  and that the input rectangles are allowed to be rotated. The problems considered in this paper are defined as follows.

**Problem 1 [Minimum area grid packing (MAGP)]** *Given a set  $\mathcal{S}$  of  $n$  rectangles find a grid packing of  $\mathcal{S}$  that minimizes the total area of the grid.*

We also consider several interesting variants of the problem, for example:

**Problem 2 [Minimum area  $k$ -grid packing (MA $k$ GP)]** *Given a set  $\mathcal{S}$  of  $n$  rectangles and an integer  $k \leq n$  compute a minimum area grid packing containing at least  $k$  rectangles of  $\mathcal{S}$ .*

**Problem 3 [Minimum area grid packing with bounded aspect ratio (MAGPAR)]** *Given a set  $\mathcal{S}$  of  $n$  rectangles and a real number  $\mathcal{R}$ , compute a minimum area grid packing whose bounding box aspect ratio is at most  $\mathcal{R}$ .*

**Problem 4 [Maximum wafer packing (MWP)]** *Given a set of  $n$  rectangles  $\mathcal{S}$  and a rectangular region  $\mathcal{A}$  compute a grid packing of  $\mathcal{S}' \subseteq \mathcal{S}$  on  $\mathcal{A}$  of maximal size.*

**Problem 5 [Minimum number of wafers (MNWP)]** *A plate is a pre-specified rectangular region. Given a set of  $n$  rectangles  $\mathcal{S}$  compute a grid packing of  $\mathcal{S}$  onto a minimal number of plates.*

Weighted variants of the problem is also studied. In the weighted case each rectangle  $r \in \mathcal{S}$  also has a weight, denoted  $w(r)$ , associated to it.

More problems related to wafer-packing can be found in [5]. A problem that is similar to grid packing is the tabular formatting problem [17,20]. In the most basic tabular formatting problem, one is given a set of rectangular entries and the aim is to construct a table containing the entries in a given order for each row and column. Shin *et al.* [17] suggested three different objective functions that evaluate the quality of a tabular layout: minimal diameter, minimal area, and minimal white space. Therefore we also consider the following problem:

**Problem 6 [Minimum diameter grid packing (MDGP)]** *Given a set  $\mathcal{S}$  of  $n$  rectangles find a grid packing of  $\mathcal{S}$  that minimizes the diameter of the grid.*

A similar problem, with the exception that the rectangles cannot be rotated, was considered by Beach [4] under the name "Random Pack". Beach showed that Random Pack is strongly  $\mathcal{NP}$ -hard, and so it partly corresponds to our  $\mathcal{NP}$ -hardness result in Theorem 5.

Another problem similar in flavor to the grid packing problem is the classical 2-dimensional bin packing problem, where one is given a set of  $n$  rectangles and an unlimited number of identical rectangular bins and the objective is to allocate all the items to the minimum number of bins. The problem has a rich history and many variants have been considered, see for example [6,7,13,14,18].

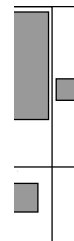


Fig. 1. Input is a set of rectangles  $\mathcal{S}$ . Output a grid packing of  $\mathcal{S}$

Our main result is a polynomial time approximation scheme (PTAS) for the MAGP-problem and some of its variants (Problems 2, 3 and 6). We also show how the algorithm can be modified to produce approximative solutions for Problems 4 and 5. Surprisingly, if the value of  $\varepsilon$  is a large enough constant the approximation algorithms will run in linear time.

The approximation algorithms all build upon the same ideas. The main idea is that for every possible grid  $\mathcal{G}$  that encloses  $\mathcal{S}$ , there exists a grid  $\mathcal{G}'$  that can be uniquely coded using only  $\mathcal{O}(\log n)$  bits such that the area of  $\mathcal{G}'$  is at most a factor  $(1 + \varepsilon)$  larger than the area of  $\mathcal{G}$ . Now, let  $\mathcal{F}$  be the family of these grids that can be uniquely coded using  $\mathcal{O}(\log n)$  bits. It trivially follows that there is only a polynomial number of grids in  $\mathcal{F}$ . Hence, every grid  $\mathcal{G}'$  in  $\mathcal{F}$  can be generated and tested. The test is performed by computing a maximal packing of  $\mathcal{S}$  into  $\mathcal{G}'$ , which in turn is done by transforming the problem into an instance for the max-flow problem, i.e., given a directed graph with a capacity function for each edge, find the maximum flow through the graph. To obtain a PTAS one uses  $\mathcal{O}(\log_{1+\varepsilon} n)$  bits for coding a grid in  $\mathcal{F}$ . In a similar way only  $\log n/2$  bits are used to obtain a linear time approximation algorithm, however the approximation factor will be quite large in this case. More details about  $\mathcal{F}$  are given in Section 3 together with two important properties.  $\mathcal{F}$  is a member of the family of so-called  $(\alpha, \beta, \gamma)$ -restricted grids, which are also described in Section 3. Then, in Section 4 we show how a grid is tested by reducing the problem to a min-cost max-flow problem. The main theorem is also stated in this section. In Section 5 we consider variants of the MAGP-problem, and finally, in Section 6 we show some hardness results. Our model of computation is the traditional algebraic computation tree model. In our algorithms we will for simplicity of writing use the notation  $\lceil \mathcal{R} \rceil$  to denote the ceiling of a real number  $\mathcal{R}$ , but the time to compute it will be assumed to be  $\mathcal{O}(\log_2 \mathcal{R})$ .

### 1.1 Approximability preserving simplifications

In most practical applications we believe that the side lengths of the input rectangles belongs to the interval  $[1, n^c]$ , for some constant  $c$ . Below we will also show that for most of the problems considered in this paper the side lengths of the input rectangles can be scaled such that the scale lengths lie in the interval  $[1, n^c]$  without affecting the approximation results by more than a factor  $(1 + \varepsilon)$ , for any constant  $\varepsilon > 0$ . Set  $c' = c/2$ .

Let  $h$  be the length of the longest side of any rectangle in  $\mathcal{S}$  and, let  $w$  be the length of the longest short side of any rectangle in  $\mathcal{S}$ . Let  $\mathcal{G}$  denote an optimal grid packing, and let  $H$  and  $W$  denote the height and width of  $\mathcal{G}$ . Note that the area of  $\mathcal{G}$  is bounded from below by  $(h \cdot w)$  and from above by  $(h \cdot nw)$ . We will have two cases, depending on the ratio  $h/w$ .

In the case when  $h/w < n^{c'}$  we may assume that the width and height of each rectangle  $r \in \mathcal{S}$  is in the interval  $[1, n^{2c'}]$ . Note that  $w > n^{c'}$ , since  $h = n^{2c'}$ . We argue that this assumption can be made in this case without loss of generality with respect to our approximation results. Since the area is approximated we may assume that the shortest side of any input rectangle has length at least 1. If not, the side is expanded such that it has length 1. Note that the side lengths of any solution will expand by at most  $n$  which is at most a factor  $(1 + n^{1-c'})$  larger than an optimal solution and therefore negligible for the approximation algorithms we consider in this paper. Hence, we may assume that the sides of the input rectangles have length within the interval  $[1, n^{2c'}]$ .

In the case when  $h/w \geq n^{c'}$ , we will show that the orientation of the rectangles in  $\mathcal{S}$  can be fixed, i.e., all rectangles are packed “standing”. For simplicity we assume that  $H \geq W$ . Let  $L$  be the set of rectangles that are lying down in the optimal packing  $\mathcal{G}$ , and denote by  $h'$  the length of the longest long side of any rectangle in  $L$ . If  $L$  is empty then the claim holds, hence, we may assume that  $L$  contains at least one rectangle. We will have two subcases:

- (1) If  $h' > h/n^2$  then the area of  $\mathcal{G}$  is at least  $h \times h' \geq h^2/n^2$  which is greater than  $h \times w$ , thus, we have a contradiction since  $area(\mathcal{G}) \leq h \times w = h^2/n^{c'}$ .
- (2) If  $h' \leq h/n^2$  then we can place all the rectangles in  $L$  standing on top of the optimal grid. Obviously the width is the same as  $\mathcal{G}$  and the height increases by at most  $n \cdot h' < h/n$ . Thus, the area of the new grid is bounded by  $(H + h/n) \times W \leq (1 + \varepsilon)H \times W$ .

Hence, in the case when  $h/w \geq n^{c'}$  we may fix the orientation of the input rectangles (standing). Having fixed the orientation, we may without loss of generality create a new scale for horizontal lengths. In this new scale we can again define the longest width of a “standing” rectangle to be equal to  $n^c$ . In this way, using the same arguments as previously, we may assume that all sides of all rectangles have length in the interval  $[1, n^c]$ .

Similarly, we may assume that all rectangles have weight in the interval  $[1, n^c]$ . Let  $\max_{r \in \mathcal{S}} w(r) = n^c$ ; it holds that all rectangles with weight less than 1 can be discarded since their combined weight sums up to at most  $n$ , which is at most  $n^{1-c}$  of the weight of an optimal solution and hence negligible.

Note that the above simplification cannot be applied to Problem 2 and 4.

## 2 The approximation algorithm

The approximation algorithm is simple and straight-forward, therefore we here give the global structure. The two non-trivial steps, lines 11 and 12, will be

described in detail in Sections 3 and 4 respectively. The last step, `PACKINTOGRID`, is obtained by slightly modifying the procedure `TESTGRID`. As input to the algorithm we will be given a set  $\mathcal{S}$  of  $n$  rectangles and a real value  $\varepsilon' > 0$ .

**Algorithm** `GRIDPACK`( $\mathcal{S}, \varepsilon'$ )

1.  $c$  is calculated as defined in Section 1
2.  $bestVal \leftarrow \infty$ ,  $n \leftarrow |\mathcal{S}|$ ,  $\alpha = \beta = \sqrt{1 + \varepsilon'}$ ,  $\gamma = \frac{1}{\sqrt{1 + \varepsilon'} - 1}$
3. **for** each  $1 \leq i, j \leq \log_{\alpha} n^c$  **do**
4.      $\mathcal{S}_{i,j} \leftarrow \emptyset$
5.     **for** each  $r \in \mathcal{S}$  **do**
6.          $i \leftarrow \lceil \log_{\alpha} \text{width}(r) \rceil$
7.          $j \leftarrow \lceil \log_{\alpha} \text{height}(r) \rceil$
8.          $\mathcal{S}_{i,j} \leftarrow \mathcal{S}_{i,j} \cup \{r\}$
9.     **end**
10.    **for**  $k \leftarrow 1$  **to**  $n^{2c \cdot f(\alpha, \beta, \gamma)}$  **do**
11.        $\mathcal{G} \leftarrow \text{GENERATEGRID}(\alpha, \beta, \gamma, k, n, c)$
12.        $val \leftarrow \text{TESTGRID}(\mathcal{G}, \{\mathcal{S}\}_{1 \leq i, j \leq \log_{\alpha} n^c}, \alpha, \beta, \gamma)$
13.       **if**  $val < bestVal$  **then**
14.            $bestVal \leftarrow val$  and  $bestGrid \leftarrow \mathcal{G}$
15.     **end**
16.    **Output** `PACKINTOGRID`( $\mathcal{S}, bestGrid$ )

The initialization is performed on lines 1 to 4. On lines 5–9, the rectangles are partitioned into groups in such a way that a rectangle  $r \in \mathcal{S}$  belongs to  $\mathcal{S}_{i,j}$  if and only if the width of  $r$  is between  $\alpha^{i-1}$  and  $\alpha^i$ , and the height of  $r$  is between  $\alpha^{j-1}$  and  $\alpha^j$ . Lines 1-9 obviously run in linear time. Next, a sequence of grids  $\mathcal{G}$  are produced in a loop of lines 10-15. They are members of the family of so-called  $(\alpha, \beta, \gamma)$ -restricted grids which is described in Section 3. (This family consists of  $n^{(2c \cdot f(\alpha, \beta, \gamma))}$  grids, where  $f(\alpha, \beta, \gamma) = (\log(\alpha\beta\gamma))/(\log \alpha \log \beta)$ .)

The generated grid is tested and the weight of an approximative grid packing of  $\mathcal{S}$  into the grid  $\mathcal{G}$  is computed. If the grid packing is better than the previously tested grids then  $\mathcal{G}$  is saved as the best grid tested so far. Finally, when all grids in  $\mathcal{F}$  have been generated and tested a call to `PACKINTOGRID` performs a grid packing of  $\mathcal{S}$  into the best grid found. This procedure is a simple modification of the `TESTGRID`-step and it is briefly described in Theorem 1 and Lemma 2.

### 3 The family $\mathcal{F}$ of $(\alpha, \beta, \gamma)$ -restricted grids

The aim of this section is to define the family  $\mathcal{F}(\alpha, \beta, \gamma, n, c)$ , or  $\mathcal{F}$  for short, of  $(\alpha, \beta, \gamma)$ -restricted grids and prove two properties about  $\mathcal{F}$ . However, before

the properties can be stated we need the following definition. A grid  $G_1$  is said to *include* a grid  $G_2$  if every possible set of rectangles that can be grid packed into  $G_2$  also can be grid packed into  $G_1$ .  $\mathcal{F}$  has the following two properties.

- (1) For every grid  $G$  that encloses  $\mathcal{S}$ , and hence has at most  $n$  rows and  $n$  columns, there exists a grid  $\mathcal{G} \in \mathcal{F}$  that includes  $G$  and whose width and height is at most a factor  $\left(\frac{\alpha^2\beta\gamma}{\alpha\gamma-1}\right)$  times larger than the width and height of  $G$ , and
- (2)  $\#\mathcal{F} \leq n^{(2^c \cdot f(\alpha, \beta, \gamma))}$  where  $f(\alpha, \beta, \gamma) = \frac{\log(\alpha\beta\gamma)}{\log \alpha \log \beta}$ .

The definition of an  $(\alpha, \beta, \gamma)$ -restricted grid is somewhat complicated, therefore we choose to describe this step by step.

A trivial observation is that two grid-packings are equivalent if the one can be transformed to the other by exchanging the order of rows and/or the columns. Hence we may assume that the columns are ordered with respect to decreasing width from left to right and that the rows are ordered with respect to decreasing height from top to bottom. This ordering will be assumed throughout the paper.

### 3.1 Size of an $(\alpha, \beta, \gamma)$ -restricted grid

Consider an arbitrary grid  $G$  and let  $\alpha$  be a real constant greater than 1. An  $\alpha$ -restricted grid is a grid where the width and height of each cell in the grid is an integral power of  $\alpha$ .

**Observation 1** *For any grid  $G$  there exists an  $\alpha$ -restricted grid  $\mathcal{G}$  that includes  $G$  and whose width and height is at most a factor  $\alpha$  longer than the width and height of  $G$ .*

**PROOF.** The observation is straight-forward since the width and height of each cell of  $G$  can increase by at most a factor  $\alpha$  to make the grid  $\alpha$ -restricted.  $\square$

Let  $G$  be a  $\alpha$ -restricted grid. If the number of columns/rows of each size  $\beta^i$ , for some integer  $i$ , then  $G$  is an  $(\alpha, \beta)$ -restricted grid.

**Observation 2** *For any  $\alpha$ -restricted grid  $G$  there exists an  $(\alpha, \beta)$ -restricted grid  $\mathcal{G}$  that includes  $G$  and whose width and height is at most a factor  $\beta$  greater than the width and height of  $G$ .*

**PROOF.** The observation is straight-forward since  $\mathcal{G}$  can be obtained from  $G$  by increasing the number of columns/rows by at most a factor  $\beta$ , hence the observation follows.  $\square$

Note, as  $\beta^i$  may not be an integer, the more formal definition of  $(\alpha, \beta)$ -restricted grid would be as above, but with  $\beta^i$  replaced by  $\lfloor \beta^i \rfloor$ . However, the above observation holds for both cases and for simplicity of writing  $\beta^i$  will be assumed to be an integer.

The columns/rows in an  $\alpha$ -restricted grid of width/height  $\alpha^i$  are said to have column/row size  $i$ . A grid  $G$  is said to be  $\gamma$ -monotone if the number of columns (rows) of size  $i$  is greater than or equal to  $1/\gamma > 0$  times the number of columns (rows) of size  $i + 1$  for every  $i$ .

**Observation 3** *Let  $\gamma$  be a constant greater than or equal to 1. For any  $(\alpha, \beta)$ -restricted grid  $G$  there exists a  $\gamma$ -monotone  $(\alpha, \beta)$ -restricted grid  $\mathcal{G}$  ( $(\alpha, \beta, \gamma)$ -restricted grid for short) that includes  $G$  and whose width and height is at most a factor  $(\gamma\alpha)/(\gamma\alpha - 1)$  greater than the width and height of  $G$ .*

**PROOF.** Since the number of columns is decreasing with at most a factor  $\gamma$  and since the width of the columns decrease with a factor  $\alpha$  for each size we obtain that the worst-case can be bounded by a geometric series,  $\sum_{i=1}^{\infty} 1/(\gamma\alpha)^i < (\gamma\alpha)/(\gamma\alpha - 1)$ . The observation follows.  $\square$

Putting together the observations immediately gives the following corollary.

**Corollary 1** *For any grid  $G$  there exists an  $(\alpha, \beta, \gamma)$ -restricted grid  $\mathcal{G}$  that includes  $G$  and whose width and height is at most a factor  $\left(\frac{\alpha^2\beta\gamma}{\alpha\gamma-1}\right)$  greater than the width and height of  $G$ .*

Most often we do not need the actual grid, instead we are interested in the number of cells in the grid of a certain size. That is, the grid  $\mathcal{G}$  is represented by a  $\lceil \log_{\alpha} n^c \rceil \times \lceil \log_{\alpha} n^c \rceil$  integer matrix, where  $\mathcal{G}[i, j]$  stores the number of cells in  $\mathcal{G}$  of width  $\alpha^i$  and, height  $\alpha^j$ . We call this a matrix representation of a grid.

### 3.2 Size of the family $\mathcal{F}$ of $(\alpha, \beta, \gamma)$ -restricted grids

We are now ready to formally define the family of grids to be considered by the algorithm.

**Definition 2** *Consider  $c > 0$  and  $\alpha, \beta, \gamma > 1$  as well as a positive integer  $n$ . By  $\mathcal{F}(\alpha, \beta, \gamma, n, c)$ , or  $\mathcal{F}$  for short, we denote the set of all  $(\alpha, \beta, \gamma)$ -restricted grids where both the size of each row and the size of each column is upper bounded by  $\log_{\alpha} n^c$ .*

Using Corollary 1 it is now straight-forward to see that Property 1 holds for  $\mathcal{F}$ . Next we show that the second property holds for  $\mathcal{F}$ , i.e., the number of

grids that are members of  $\mathcal{F}$  is at most  $n^{(2c \cdot f(\alpha, \beta, \gamma))}$ . It will be a constructive proof where it will be shown that every grid in  $\mathcal{F}$  can be coded uniquely with  $2(\log_\alpha n^c(1 + \log_\beta \gamma) + \log_\beta n^2)$  bits. Hence, producing all words of length  $2(\log_\alpha n^c(1 + \log_\beta \gamma) + \log_\beta n^2)$  will also generate all members of  $\mathcal{F}$ .

Assume that we are given a member  $f \in \mathcal{F}$  and that  $f$  has  $r_j$  rows of size  $j$ . Recall that  $r_j$  is an integral power of  $\beta$ . The idea of the scheme is as follows, see also algorithm CODINGROWS below. The bit string, denoted  $S$ , is built incrementally. Consider a generic step of the algorithm. Assume that the bit string, denoted  $S_{j+1}$  has been built for all the row sizes greater than  $j$  and that the number of rows of size  $(j+1)$  is  $r_{j+1}$ . Initially  $S_{\log_\alpha n^c}$  is the empty string. Consider the row size  $j$ . We will have two cases, either  $r_j \leq (r_{j+1}/\gamma)$  or  $r_j > (r_{j+1}/\gamma)$ . In the first case, add ‘1’ to  $S_{j+1}$  to obtain  $S_j$ . In the latter case, when  $r_j > (r_{j+1}/\gamma)$ , add  $(\log_\beta r_j - \max((\log_\beta r_{j+1} - \lceil \log_\beta \gamma \rceil), 0))$  zeros followed by a ‘1’ to  $S_{j+1}$  to obtain  $S_j$ . Decrease the value of  $j$  and continue the process until  $j = 0$ , and hence,  $S_0 = S$ .

The algorithm above describes how to generate the rows, the coding for the columns is done in the same way. The following observation proves that property 2 holds for the family of  $(\alpha, \beta, \gamma)$ -restricted grids.

**Observation 4** *The length of  $S$  is  $2(\log_\alpha n^c(1 + \log_\beta \gamma) + \log_\beta n^2)$ .*

**PROOF.** Consider the code for the rows. There are  $\log_\alpha n^c$  different row sizes and, therefore, also at most  $\lceil \log_\alpha n^c \rceil$  many ‘1’-s. The total number of rows is at most  $n^2$ , hence the number of ‘0’-s is bounded by  $\log_\alpha n^c \cdot \log_\beta \gamma + \log_\beta n^2$ .  $\square$

**Algorithm CODINGROWS** $(\alpha, \beta, \gamma, n, c)$

1.  $size \leftarrow 2(\log_\alpha n^c(1 + \log_\beta \gamma) + \log_\beta n^2)$
2.  $S \leftarrow array[1..size]$ ,  $r_{\lceil \log_\alpha n^c \rceil + 1} \leftarrow 0$ ,  $index \leftarrow 1$
3. **for** each row size  $j = \lceil \log_\alpha n^c \rceil$  **downto** 1 **do**
4.      $\#Rows \leftarrow \log_\beta r_j - \max((\log_\beta r_{j+1} - \lceil \log_\beta \gamma \rceil), 0)$
5.     **for**  $k = 1$  **to**  $\#Rows$  **do**
6.          $S[index] \leftarrow '0'$
7.          $index \leftarrow index + 1$
8.     **end**
9.      $S[index] \leftarrow '1'$
10.      $index \leftarrow index + 1$
11.    **end**
12.    **while**  $index \leq 2(\log_\alpha n^c(1 + \log_\beta \gamma) + \log_\beta n^2)$  **do**
13.          $S[index] \leftarrow '0'$
14.          $index \leftarrow index + 1$
15.    **end**
16.    **Output**  $S$

### 3.3 GenerateGrid

On line 11 in algorithm GRIDPACK the procedure GENERATEGRID is called with the parameters  $\alpha, \beta, \gamma$  and  $k$ , where  $k$  is a positive integer  $\leq n^{(2c \cdot f(\alpha, \beta, \gamma))}$  and hence its binary representation, denoted  $B(k)$ , has length  $2(\log_\alpha n^c(1 + \log_\beta \gamma) + \log_\beta n^2)$ . Note that for simplicity we chose to generate all bit strings of length at most  $2(\log_\alpha n^c(1 + \log_\beta \gamma) + \log_\beta n^2)$  and then decide in the procedure GENERATEGRID if it corresponds to an  $(\alpha, \beta, \gamma)$ -restricted grid or not. Alternatively, one could generate only valid bit strings. We obtain the following corollary:

**Corollary 2** *Given a bit string  $B$  of length  $2(\log_\alpha n^c(1 + \log_\beta \gamma) + \log_\beta n^2)$  one can in time  $\mathcal{O}(\log^2 n)$  construct the unique matrix representation of the corresponding  $(\alpha, \beta, \gamma)$ -restricted grid, or decide that there is no corresponding  $(\alpha, \beta, \gamma)$ -restricted grid.*

**PROOF.** It is easily seen that deciding if  $B$  corresponds to an  $(\alpha, \beta, \gamma)$ -restricted grid can be done in linear time w.r.t. the length of  $B$ . If  $B$  corresponds to a  $(\alpha, \beta, \gamma)$ -restricted grid the matrix representation of the grid can easily be computed in time  $\mathcal{O}(\log_\alpha^2 n + \log_\beta^2 n)$  since the number of columns/rows of each size is obtained by traversing the string, hence the corollary follows.  $\square$

## 4 Testing an $(\alpha, \beta, \gamma)$ -restricted grid

In the previous section we showed a simple method to generate all possible  $(\alpha, \beta, \gamma)$ -restricted grids. For the approximation algorithm, shown in Section 2, to be efficient we need a way to pack a maximal number of rectangles of  $\mathcal{S}$  into the grid. As input we are given a matrix representation of an  $(\alpha, \beta, \gamma)$ -restricted grid  $\mathcal{G}$ , and a set  $\mathcal{S}$  of  $n$  rectangles partitioned into groups  $\mathcal{S}_{i,j}$  depending on their width and height. Let  $\mathcal{C}_{p,q}$  denote the number of cells in  $\mathcal{G}$  that have width  $\alpha^p$  and height  $\alpha^q$ . We will give an exact algorithm for the problem by reformulating it as a max-flow problem. The problem could also be solved by reformulating it as a matching problem but in the next section we will show that the max-flow formulation can be extended to the weighted case. The max-flow problem is as follows. Given a directed graph  $G(V, E)$  with capacity function  $u(e)$  for each edge  $e$  in  $E$ . Find the maximum flow  $f$  through  $G$ .

The flow network  $\mathcal{F}_{\mathcal{S}, \mathcal{G}}$  corresponding to a grid  $\mathcal{G}$  and a set of rectangles  $\mathcal{S}$  contains four levels, numbered from top-to-bottom, and will be constructed level-by-level. In figure 3, the weighted variant of the network is shown. The number of nodes on level  $i$ ,  $1 \leq i \leq 4$ , is denoted  $m_i$ .

- Level 1.** At the top level there is one node, the source node  $\nu^{(1)}$ .
- Level 2.** At level 2 there are  $\log_\alpha^2 n^c$  nodes. A node  $\nu_{i,j}^{(2)}$  at level 2 represents the group  $\mathcal{S}_{i,j}$ . For each node  $\nu_{i,j}^{(2)}$ , there is a directed edge from  $\nu^{(1)}$  to  $\nu_{i,j}^{(2)}$ . The capacity of this edge is equal to the number of rectangles in  $\mathcal{S}$  that belong to  $\mathcal{S}_{i,j}$ .
- Level 3.** At level 3 there are also  $\log_\alpha^2 n^c$  nodes. A node  $\nu_{p,q}^{(3)}$  on level 3 represents the set of cells in  $\mathcal{C}_{p,q}$ . For each node  $\nu_{p,q}^{(3)}$  there is a directed edge from node  $\nu_{i,j}^{(2)}$  to node  $\nu_{p,q}^{(3)}$  if and only if  $p \geq i$  and  $q \geq j$  (or  $q \geq i$  and  $p \geq j$ ), i.e., if a rectangle in  $\mathcal{S}_{i,j}$  can be packed into a cell in  $\mathcal{C}_{p,q}$ . All edges from level 2 to level 3 have capacity  $n$ .
- Level 4.** The bottom level only contains one node, the sink  $\nu^{(4)}$ . For every node  $\nu_{p,q}^{(3)}$  on level 3 there is a directed edge from  $\nu_{p,q}^{(3)}$  to  $\nu^{(4)}$ . The capacity of this edge is equal to the number of cells in  $\mathcal{G}$  that belongs to  $\mathcal{C}_{p,q}$ .

The following observation is straight-forward.

**Observation 5** *The maximal grid packing of  $\mathcal{S}$  into  $\mathcal{G}$  has size  $k$  if and only if the max flow in the flow network is  $k$ .*

**PROOF.** Every unit flow corresponds to a rectangle and since the flow is  $k$  it implies that every rectangle in  $\mathcal{S}$  has been matched to a cell. It is easily seen that a rectangle can only be matched to a cell it can fit into, hence if the flow is  $k$  there exists a grid packing of size  $k$  of  $\mathcal{S}$  into  $\mathcal{G}$ .

The ‘only if’ statement is obvious since if there exists a grid packing there must exist a  $k$ -matching between the rectangles and the cells, and if this is true there must be a maximal flow of value  $k$ .  $\square$

In 1998 Goldberg and Rao [10] presented an algorithm for the maximum flow problem with running time  $\mathcal{O}(N^{2/3}M \log(N^2/M) \log U)$ . If we apply their algorithm to the flow network we obtain the following lemma.

**Lemma 1** *Given a matrix representation of an  $(\alpha, \beta, \gamma)$ -restricted grid  $\mathcal{G}$  and a set  $\mathcal{S}$  of  $n$  rectangles partitioned into the groups  $\mathcal{S}_{i,j}$  w.r.t. their width and height. (1) The size of an optimal packing of  $\mathcal{S}$  in  $\mathcal{G}$  can be computed in time  $\mathcal{O}(\log^{19/3} n)$ . (2) An optimal packing can be computed in time  $\mathcal{O}(\log^{19/3} n + k)$ , where  $k$  is the number of rectangles in the optimal grid packing of  $\mathcal{S}$  in  $\mathcal{G}$ .*

**PROOF.** The lemma follows by plugging in the different values for  $N$ ,  $M$  and  $U$  in the above mentioned algorithm by Goldberg and Rao. We have that the number of nodes ( $N$ ) is  $\Theta(\log_\alpha^2 n^c)$ , the number of edges ( $M$ ) is  $\Theta(\log_\alpha^4 n^c)$ , and finally, the integral arc capacities ( $U$ ) are in the interval  $[1..n]$ .  $\square$

As a result we obtain the first grid packing theorem.

**Theorem 1** *Algorithm GRIDPACK is a  $(1 + \varepsilon)$ -approximation algorithm for the MAGP-problem with running time  $\mathcal{O}(n^{f(\sqrt{1+\varepsilon/7})} \log^{19/3} n + n)$ , where  $f(\chi) = \frac{2c \log(\chi/(\sqrt{\chi}-1))}{\log^2 \chi}$ .*

**PROOF.** Corollary 1 guarantees the stated approximation ratio, since it holds that  $\left(\frac{\alpha^2 \beta \gamma}{\alpha \gamma - 1}\right)^2 = (1 + \varepsilon/7)^3 < (1 + \varepsilon)$ . The time-complexity of the approximation algorithm is dominated by testing if the set of rectangles can be grid packed into a grid  $\mathcal{G}$ . The total time for this step is the number of tested grids times the time to test one grid, hence,  $\mathcal{O}(n^{f(\chi)} \log^{19/3} n + n)$ .

The final packing is easily computed from the max-flow and hence can be done in time linear with respect to the number of rectangles in the packing.  $\square$

Even though the expression for the running time in the above theorem looks somewhat complicated it is not hard to see that by choosing the value of  $\varepsilon$  appropriately we obtain that, algorithm GRIDPACK is a PTAS for the MAGP-problem, and if  $\varepsilon$  is set to be a large constant GRIDPACK produces a grid packing that is within a constant factor of the optimal in linear time.

## 5 Applications and extensions

The approximation algorithm presented above can be extended and generalized to variants of the basic grid packing problem. Below we show how to extend the algorithm to these variants.

By performing some small modifications to the procedure TESTGRID we will obtain the following corollary from Theorem 1. Let  $f(\chi) = \frac{2c \log(\chi/(\sqrt{\chi}-1))}{\log^2 \chi}$  and let  $g(\alpha, \beta, \gamma) = \left(\frac{\alpha^2 \beta \gamma}{\alpha \gamma - 1}\right)$ .

**Corollary 3** *Given an instance of the MNWP-problem, one can compute an  $O(1)$ -approximation in linear time provided that the optimal solution uses  $O(1)$  wafers.*

**PROOF.** Let  $\mu$  be the number of wafers used in the optimal solution of the MNWP problem. The approximation algorithm becomes as follows. First, compute a  $(1 + \varepsilon)$ -approximation for the MAGP-problem using the GRIDPACK algorithm and an  $\varepsilon$  large enough for the running time to become linear. Let  $G$  be the corresponding grid of the solution produced. Next, create a second grid  $\mathcal{Q}$ , where each cell has the same size and shape as  $\mathcal{A}$ , and place  $\mathcal{Q}$  on top of  $G$ . The number of cells in  $\mathcal{Q}$  is minimized as to still cover all of  $G$ , where it is straightforward to see that  $\mathcal{Q}$  will contain  $\mathcal{O}(\mu^2)$  cells. Further, let  $q$  be an arbitrary cell in  $\mathcal{Q}$  and let  $G(q)$  be the subgrid of  $G$  whose cells lie entirely within  $q$  or is intersecting the right or bottom side of  $q$ . The subgrid  $G(q)$  is

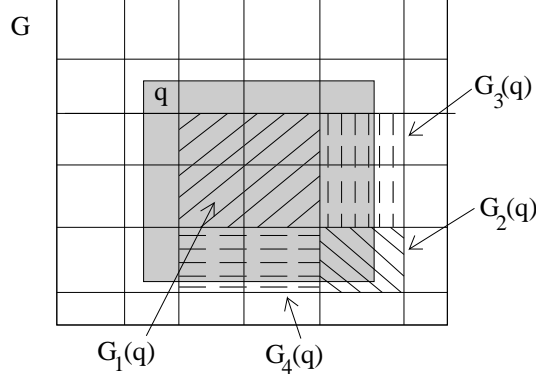


Fig. 2. Illustration of the proof of Corollary 3

now partitioned into four smaller grids, denoted  $G_1(q), \dots, G_4(q)$ , as shown in Fig. 2. The first grid contains all cells of  $G(q)$  that lie entirely within  $q$ . Grid  $G_2(q)$  contains all cells of  $G(q)$  intersecting the bottom right corner of  $q$ , and finally  $G_3(q)$  and  $G_4(q)$  are the grids induced by the cells in  $G(q) \setminus (G_1(q) \cup G_2(q))$  intersecting the bottom and right side, respectively, of  $q$ . For each  $q$  we pack each smaller grid  $G_1(q), \dots, G_4(q)$  on a separate region  $\mathcal{A}$ . Obviously such a packing is possible for  $G_1$ , and also for  $G_2(q), \dots, G_4(q)$  as they each consist of a single row/column of width/height smaller than the width/height of  $\mathcal{A}$ . Thus, all of  $\mathcal{S}$  may be packed on  $\mathcal{O}(\mu^2)$  regions  $\mathcal{A}$  and the corollary follows.  $\square$

Note that the approximation factor for MNWP can be made smaller if we adapt the above method using an  $\epsilon$  close to zero, but this would imply that the running time, although still polynomial, is not linear any more.

**Corollary 4** *Given a set  $\mathcal{S}$  of rectangles and a real positive value  $\epsilon$  one can compute a  $(1+\epsilon)$ -approximation of the MDGP-problem in time  $\mathcal{O}(n^{f(\sqrt{1+\epsilon/3})} \log^{19/3} n + n)$ .*

**PROOF.** Consider an optimal grid  $G$  with height  $h$  and width  $w$ . There exists an  $(\alpha, \beta, \gamma)$ -restricted grid  $\mathcal{G}$  that includes  $G$  and whose height and width is at most a factor  $g(\alpha, \beta, \gamma)$  greater than  $h$  and  $w$  respectively, hence its diameter is at most a factor  $g(\alpha, \beta, \gamma)$  greater than the optimal.  $\square$

The proof of Corollary 4 can easily be modified to hold for Corollaries 5-7.

Let  $\psi(\mathcal{S}, \mathcal{R})$  denote the area of minimum area grid packing of  $\mathcal{S}$  with aspect ratio  $\mathcal{R}$ .

**Corollary 5** *Given a set  $\mathcal{S}$  of rectangles, a real number  $\mathcal{R}$  and a real positive value  $\epsilon$  one can compute a grid packing for the MAGPAR-problem of area  $(1+\epsilon) \cdot \psi(\mathcal{S}, \mathcal{R})$  and aspect ratio at most  $(1+\epsilon)\mathcal{R}$  in time  $\mathcal{O}(n^{f(\sqrt{1+\epsilon/3})} \log^{19/3} n + n)$ .*

**Corollary 6** *Given a set  $\mathcal{S}$  of  $n$  rectangles, an integer  $k \leq n$  and a real positive value  $\varepsilon$ , where the height and width of each rectangle  $r \in \mathcal{S}$  is in the range  $[1, n^c]$ , one can compute a  $(1 + \varepsilon)$ -approximation of the MAkGP-problem in time  $\mathcal{O}(n^{f(\sqrt{1+\varepsilon/7})} \log^{19/3} n + n)$ .*

Let  $\kappa(\mathcal{S}, \mathcal{A})$  denote the cardinality of a maximum grid packing of  $\mathcal{S}$  onto  $\mathcal{A}$ . Let  $\mathcal{A}'$  be a rectangular region such that the sides of  $\mathcal{A}'$  are a factor  $(1 + \varepsilon)$  longer than the sides of  $\mathcal{A}$ .

**Corollary 7** *Given a set  $\mathcal{S}$  of  $n$  rectangles, a rectangular region  $\mathcal{A}$  and a real positive value  $\varepsilon$ , one can compute a packing for the MWP-problem of  $\mathcal{S}$  onto  $\mathcal{A}$  of size at least  $\kappa(\mathcal{S}, \mathcal{A}')$  in time  $\mathcal{O}(n^{f(\sqrt{1+\varepsilon/3})} \log^{19/3} n + n)$ .*

### 5.1 Extending the test algorithm

The problems considered below are weighted variants of the grid packing problem, hence every rectangle  $r \in \mathcal{S}$  also has a weight  $w(r)$ . To be able to apply the above algorithm to the weighted case we need a way to pack a maximal number (weight) of rectangles in  $\mathcal{S}$  into the grid, or at least approximate the weight of a maximal packing. As input we are given a matrix representation of a  $(\alpha, \beta, \gamma)$ -restricted grid  $\mathcal{G}$ , and a set  $\mathcal{S}$  of  $n$  rectangles partitioned into groups  $\mathcal{S}_{i,j}$  depending on their width and height. We will give a  $\tau$ -approximation algorithm for the problem by reformulating it as a min-cost max-flow problem, where  $\tau > 1$  is a given parameter. Hence the call to TESTGRID in algorithm GRIDPACK will now include a fifth parameter  $\tau$ .

A min-cost max-flow problem in a directed graph where each edge  $e$  has a capacity  $c(e)$  and a cost  $d(e)$  is to find the maximum flow  $f$  with a minimum cost, where the cost of a flow  $f$  is  $\sum_{e \in E} d(e) \cdot f(e)$ .

The network that will be constructed next contains four levels, numbered from top-to-bottom, and will be constructed level-by-level. A rectangle  $r$  in  $\mathcal{S}$  belongs to weight class  $W_k$  if and only if  $w(r)$  is between  $\tau^{k-1}$  and  $\tau^k$ . The number of nodes on level  $i$ ,  $1 \leq i \leq 4$ , is denoted  $m_i$  and we set  $W = \sum_{r \in \mathcal{S}} w(r)$ .

**Level 1.** At the top level there is one node, the source node  $\nu^{(1)}$ .

**Level 2.** At level 2 there are  $\log_\alpha^2 n \cdot \log_\tau n$  nodes. A node  $\nu_{i,j,k}^{(2)}$  at level 2 represents the class of rectangles in the group  $\mathcal{S}_{i,j}$  and weight class  $W_k$ . For each node  $\nu_{i,j,k}^{(2)}$ , there is a directed edge from  $\nu^{(1)}$  to  $\nu_{i,j,k}^{(2)}$ . The capacity of this edge is equal to the number of rectangles in  $\mathcal{S}$  that belongs to  $\mathcal{S}_{i,j}$  and weight class  $W_k$ . The cost per unit flow is  $W - \tau^{k-1}$ , i.e., a rectangle with greater weight is “cheaper” to send than a rectangle with low weight.

**Level 3.** At level 3 there are  $\log_\alpha^2 n^c$  nodes, one for each set  $\mathcal{C}_{p,q}$  of cells. Hence, a node  $\nu_{p,q}^{(3)}$  on level 3 represents the set of cells in  $\mathcal{C}_{p,q}$ . For each node  $\nu_{p,q}^{(3)}$  there is a directed edge from node  $\nu_{k,i,j}^{(3)}$  to node  $\nu_{p,q}^{(3)}$  if and only if  $p \geq i$  and  $q \geq j$ , i.e., the rectangles in  $\mathcal{S}_{i,j}$  can be packed into the cells in  $\mathcal{C}_{p,q}$ . All the edges from level 2 to level 3 have capacity  $n$  and zero cost.

**Level 4.** The bottom level only contains one node, the sink  $\nu^{(4)}$ . For every node  $\nu_{p,q}^{(3)}$  on level 3 there is a directed edge from  $\nu_{p,q}^{(3)}$  to  $\nu^{(4)}$  with cost 0 and capacity  $|\mathcal{C}_{p,q}|$ .

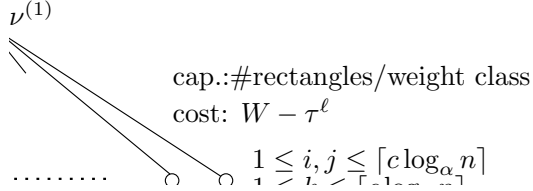


Fig. 3. Illustrating the flow network given as input to the min-cost max-flow algorithm.

We need the following observation before we can state the main lemma.

**Observation 6** *If the min-cost max-flow of  $\mathcal{F}_{\mathcal{S},\mathcal{G}}$  is  $(W-w)$  then the maximal grid packing of  $\mathcal{S}$  into  $\mathcal{G}$  has weight at most  $\tau w$ .*

**PROOF.** The min-cost max-flow of cost  $(W-w)$  is equivalent to a max cost max flow of cost  $w$ . It is not hard to see that if there exists a max flow of max cost  $w$  then there exists a maximum matching between the rectangles in  $\mathcal{S}$  and the cells of  $\mathcal{G}$  of total weight  $w$ . Since the weight of each rectangle is rounded down such that its weight is divisible by  $\tau$  it holds that the weight of a maximal packing is at most  $\tau w$ .  $\square$

We summarize this section by stating the following lemma.

**Lemma 2** *Given a matrix representation of an  $(\alpha, \beta, \gamma)$ -restricted grid  $\mathcal{G}$  and a set  $\mathcal{S}$  of  $n$  rectangles partitioned into the groups  $\mathcal{S}_{i,j}$  w.r.t. their width and height. (1) The cost of a  $\tau$ -approximate packing of  $\mathcal{S}$  onto  $\mathcal{G}$  can be computed in time  $\mathcal{O}(\log^9 n \log \log n)$ . (2) A  $\tau$ -approximate packing can be computed in time  $\mathcal{O}(\log^9 n \cdot \log \log n + k)$ , where  $k$  is the number of rectangles in the grid packing.*

**PROOF.** According to Observation 6 the solution to the min-cost max-flow problem on  $\mathcal{F}_{\mathcal{S},\mathcal{G}}$  will give an approximate solution to the maximal grid packing problem of  $\mathcal{S}$  into  $\mathcal{G}$  that is at most a factor  $\tau$  of the optimal.

Let  $N$  be the number of vertices in the network;  $M$ , the number of edges;  $U$ , the maximum value of an edge capacity; and  $C$ , the maximum value of an edge cost. The values of the parameters in the above transformed instance can be bounded as follows:  $N = \Theta(\log_\alpha^2 n \cdot \log_\tau n)$ ,  $M = \Theta(\log_\alpha^4 n \cdot \log_\tau n)$ ,  $U \leq n$  and, finally,  $C = \mathcal{O}(n^{c+1})$ . Plugging in the values in the algorithm by Ahuja *et al.* [1] with running time  $\mathcal{O}(MN \log \log U \log NC)$  gives that the max-flow min-cost problem can be solved in time  $\mathcal{O}(\log^9 n \cdot \log \log n)$ .

Finally, if we are not satisfied with an estimated cost but actually want a packing then we just pair up the rectangles with the matching cells. This is trivially done in time linear with respect to the number of rectangles in the packing.  $\square$

The minimum-cost circulation problem has a rich history and the interested reader is encouraged to read [2]. In 1989, Goldberg and Tarjan [11] showed a simple algorithm that solves the minimum-cost circulation problem in time  $\mathcal{O}(N^2 M^3 \log N)$ . There are also other types of algorithms that use capacity scaling, for example the algorithm by Edmonds and Karp [8] with a running time of  $\mathcal{O}((M \log U)(M + N \log N))$ .

## 5.2 Weighted variants

In this section we consider two weighted variants of the grid packing problem, i.e., every rectangle  $r \in \mathcal{S}$  also has a weight  $w(r)$ . Both results follow from Corollaries 1-2 and Lemma 2. Recall that  $f(\chi) = \frac{2c \log(\chi/(\sqrt{\chi}-1))}{\log^2 \chi}$ .

**Problem 7 [Minimum area weighted grid packing(MAwGP)]** *Given a set of  $n$  rectangles and a real number  $w$  compute a minimum area grid packing containing a set of rectangles of  $\mathcal{S}$  of total weight at least  $w$ .*

**Theorem 2** *Given a set  $\mathcal{S}$  of  $n$  rectangles and a real value  $\varepsilon > 0$ , one can compute a grid packing containing rectangles of total weight at least  $w$  whose area is at most  $(1 + \varepsilon)$  greater than the area of a minimum area  $(\tau w)$ -grid packing in time  $\mathcal{O}(n^{f(\sqrt{1+\varepsilon/7})} \cdot \log^9 n \cdot \log \log n + n)$ .*

**Problem 8 [Maximum weight wafer packing (MwWP)]** *Given a set of rectangles  $\mathcal{S}$  and a rectangular region  $\mathcal{A}$  compute a grid packing of  $\mathcal{S}' \subseteq \mathcal{S}$  onto  $\mathcal{A}$  of maximum weight.*

Let  $\kappa(\mathcal{S}, \mathcal{A})$  denote the weight of a maximum weight grid packing of  $\mathcal{S}$  onto  $\mathcal{A}$ . Let  $\mathcal{A}'$  be a rectangular region such that the sides of  $\mathcal{A}$  is  $(1 + \varepsilon)$  longer than the sides of  $\mathcal{A}'$ .

**Theorem 3** *Given a set  $\mathcal{S}$  of  $n$  rectangles, a rectangular region  $\mathcal{A}$  and a real value  $\varepsilon > 0$ , one can compute a grid packing of  $\mathcal{S}$  onto  $\mathcal{A}$  whose weight is at least  $\kappa(\mathcal{S}, \mathcal{A}')/\tau$  in time  $\mathcal{O}(n^{f(1+\varepsilon/7)} \cdot \log^9 n \cdot \log \log n + n)$ .*

## 6 Hardness results

In this section we show that several of the problems considered are  $\mathcal{NP}$ -hard. We start with an interesting observation:

**Observation 7** *There exists a set  $\mathcal{S}$  of  $n$  rectangles such that every optimal grid packing of  $\mathcal{S}$  on a wafer (rectangular region)  $\mathcal{A}$  induces a grid with  $\Omega(n^2)$  cells.*

**PROOF.** Let  $\mathcal{A}$  be a square with side length  $L$ , and  $\mathcal{S}$  consists of one very large square, with side length  $L - \varepsilon$  and  $n - 1$  thin long squares with side length  $L - \varepsilon$  and  $2\varepsilon/(n - 1)$ . The entire set  $\mathcal{S}$  can only be packed onto  $\mathcal{A}$  in one single way which induces  $\Omega(n^2)$  cells.  $\square$

Fig. 4. There are  $\Omega(n^2)$  shaded squares which all are empty.

**Theorem 4** *MNWP cannot be approximated within a factor of  $3/2 - \varepsilon$  for any  $\varepsilon > 0$ , unless  $\mathcal{P} = \mathcal{NP}$ .*

**PROOF.** An instance for the BINPACKING is a finite set of items  $U$ , a size  $s(u) \in \mathbb{Z}^+$  for each  $u \in U$  and a positive integer capacity  $B$ . A solution to the problem is a partition of  $U$  into disjoint sets  $U_1, \dots, U_m$  such that the sum of the items in each  $U_i$  is  $B$  or less.

The transformation is straight-forward. For each  $u \in U$  produce a rectangle  $r$  of height  $s(u)$  and width 1. Finally, set the width of a wafer to be 1, and the height to be  $B$ . Solve the Minimum number of wafer-problem. The solution

is also a solution for the BINPACKING-problem and since this cannot be approximated within a factor of  $3/2 - \varepsilon$  for any  $\varepsilon > 0$  in polynomial time [9] the same inapproximability result holds for MNWP.  $\square$

**Theorem 5** *The MWP-problem is  $\mathcal{NP}$ -hard.*

**PROOF.** Let MWP' be the decision version of MWP: given a set of rectangles  $\mathcal{S}$ , a rectangular plate  $\mathcal{A}$  and an integer  $\mathcal{L}$  can we grid pack at least  $\mathcal{L}$  rectangles of  $\mathcal{S}$  onto  $\mathcal{A}$ ?

In order to show the reduction we need the decision version of the PARTITION problem: Given integers  $a = \{a_1 \leq \dots \leq a_n\}$ , does there exist a subset  $P \subseteq I = \{1, 2, \dots, n\}$  such that  $\sum_{j \in P} a_j = \sum_{j \in I \setminus P} a_j$ ? This problem is  $\mathcal{NP}$ -complete [16].

We show that the PARTITION-problem reduces in polynomial time to the MWP' problem. Hence, given a PARTITION instance  $(a_1, \dots, a_n)$ , we create a MWP' instance  $(\mathcal{S}, \mathcal{A}, \mathcal{L})$  such that  $\mathcal{L} = n + 1$ , as shown in Fig. 4, with some minor differences. That is,  $\mathcal{A}$  is created as a square with sides  $w + \delta$ , where  $\delta = \sum_{i=1}^n a_i/2$  and  $w$  is arbitrarily chosen larger than the diagonal of a  $\delta \times \delta$  square ( $w > \sqrt{2}\delta$ ). The set  $\mathcal{S}$  is created to contain one large square with sides  $w$ , and  $n$  rectangles, one for each integer  $a_i$ , with sides  $w$  and  $a_i$ . Since we create  $n + 1$  rectangles in total, it is clear that we have a polynomial time reduction.

This reduction is valid if we can show that all rectangles of  $\mathcal{S}$  can be grid packed on exactly one plate  $\mathcal{A}$ , if, and only if, the original Partition instance is a yes instance. In order to be able to pack all of  $\mathcal{S}$  on  $\mathcal{A}$  the following grid is needed. First we need a column and a row with width and height, respectively,  $w$ . These define a cell in which the large square may be packed. As for the remaining rectangles it is clear that no such rectangle can be packed in the  $\delta \times \delta$  lower right corner, which is formed when the large square is packed. This follows since these rectangles have a side which is longer than the diagonal of this corner ( $w > \sqrt{2}\delta$ ). This means that we need either exactly one column, or exactly one row, with width, respectively height,  $a_i$ , for each rectangle corresponding to an integer  $a_i$ . Note that each such column or row defines a cell with sides  $w$  and  $a_i$ , in which the corresponding rectangle can be packed. Since  $\delta = \sum_{i=1}^n a_i/2$  it is clear that all these columns and rows can be created, and thus all of  $\mathcal{S}$  packed on  $\mathcal{A}$ , if, and only if, the original PARTITION instance is a 'yes'-instance.  $\square$

**Theorem 6** *The MAGPAR-problem is  $\mathcal{NP}$ -hard.*

**PROOF.** The proof is almost identical to the proof of Theorem 5.  $\square$

## 7 Final comments

The approximation algorithm can be generalized to  $d$  dimensions, such that the resulting grid packing has sides that are at most  $(1 + \varepsilon)$  times longer than the length of the sides of an optimal grid packing. The running time is  $\mathcal{O}(dn \log n)$  for sorting the  $d$ -dimensional rectangles into bins. The size of the coding becomes  $\mathcal{O}(d \log n)$ , which implies that the number of grids is  $n^{dc \cdot f(\alpha, \beta, \gamma)}$ . The flow network will have  $\mathcal{O}(\log^{d+1} n)$  nodes and  $\mathcal{O}(\log^{2d+1} n)$  arcs, which implies that the running time of the max-flow algorithm will be  $\mathcal{O}(\log^{3d+3} n)$ .

## 8 Acknowledgements

We thank Esther and Glenn Jennings for introducing us to the problem. We would also like to thank Bernd Gärtner for pointing us to “The table layout problem” [3,4,17,20].

Finally, we thank the anonymous referees of this journal for pointing out some errors of an older version and helping us to improve the presentation of our results.

## References

- [1] R. K. Ahuja, A. V. Goldberg, J. B. Orlin and R. E. Tarjan. Finding minimum-cost flows by double scaling. *Mathematical Programming* 53(3):243–266, 1992.
- [2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows — Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.
- [3] R. J. Anderson and S. Solti. The Table Layout Problem. In *proceedings of Symposium on Computational Geometry*, pp. 115–123, 1999.
- [4] R.J. Beach. *Setting tables and illustrations with style*. PhD thesis, Dept. of computer science, University of Waterloo, Canada, 1985.
- [5] D. H.-C. Du, I. Lin, K. C. Chang. On Wafer-Packing Problems. *IEEE Transactions on Computers*, 42(11):1382-1388, 1993.
- [6] H. Dyckhoff. Typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145-159, 1990.
- [7] H. Dyckhoff, G. Scheithauer and J. Terno. Cutting and packing, in: M. Dell’Amico, F. Maffioli and S. Martello Eds. *Annotated Bibliographies in*

Combinatorial Optimization, John Wiley & Sons, Chichester, pp. 393–412, 1997.

- [8] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.
- [9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*, W. H. Freeman and Company, San Francisco, 1979.
- [10] A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *Journal of the ACM*, 45(5):783–797, 1998.
- [11] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by cancelling negative cycles. *Journal of the ACM*, 36(4):873–886, 1989.
- [12] G. Jennings. Personal communication, 2002.
- [13] K. Jansen and G. Zhang. On rectangle packing: maximizing benefits. In *Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 204–213, 2004.
- [14] A. Lodi, S. Martello and D. Vigo. Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics* 123:379–396, 2002.
- [15] Nikko Materials, April 2004. [www.nikkomaterials.com/compound.htm](http://www.nikkomaterials.com/compound.htm)
- [16] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Mineola, NY, 1998.
- [17] K.-H. Shin, K. Kobayashi and A. Suzuki. Tafel Musik: Formatting Algorithm of Tables. *Mathematical and Computer Modelling*, 26(1):97–112, 1997.
- [18] I. Schiermeyer. Reverse-fit: a 2-optimal algorithm for packing rectangles. In *Proc. 2nd Annual European Symposium on Algorithms. Lecture Notes in Computer Science 855*, Springer-Verlag, pp. 290–299, 1994.
- [19] Virginia semiconductors, April 2004. [www.virginiasemi.com/thinfilm.cfm](http://www.virginiasemi.com/thinfilm.cfm)
- [20] X. Wang and D. Wood. Tabular formatting problems. In *Proc. of the 3rd International Workshop on Principles of Document Processing, Lecture Notes in Computer Science 1293*, Springer-Verlag, pp. 181-191, 1996.