Chips on Wafers, or Packing Rectangles into Grids

Mattias Andersson[*] Joachim Gudmundsson[†] Christos Levcopoulos[*]

In the VLSI wafer industry it is nowadays common 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 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. 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 area of a grid packing is the area of a minimal bounding box that contains all the rectangles in the grid packing.*

The general problem considered in this paper is now stated.

MAGP [Minimum area grid packing] Given a set $\mathcal{S}$ of rectangles find a minimum area grid packing of $\mathcal{S}$.

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

MAkGP [Minimum area k-grid packing] Given a set of n rectangles and an integer $k \leq n$ compute a minimum area grid packing containing at least k rectangles.

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

MWP [Maximum wafer packing] Given a set of rectangles $\mathcal{S}$ and a rectangular region $\mathcal{A}$ compute a grid packing of $\mathcal{S}' \subseteq \mathcal{S}$ on $\mathcal{A}$ such that $|S'|$ is maximized.

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

MDGP [Minimum diameter grid packing] Given a set $\mathcal{S}$ of rectangles find a minimum diameter grid packing of $\mathcal{S}$.

A problem that is similar to grid packing is the tabular formatting problem [5]. In the most basic tabular formatting problem one is given a set of rectangular table entries and the aim is to construct a table containing the entries in a given order for each row and column. A problem more similar to ours, with the exception that the rectangles cannot be rotated, was considered by Beach [3] under the name "Random Pack". Beach showed that Random Pack is strongly NP-hard.

Our main result is a PTAS for the MAGP-problem and some of its variants. Surprisingly, if the value of ε is a large enough constant the algorithms will run in linear time.

The approximation algorithms all build upon the same ideas. The main idea is that for every possible grid $\mathcal{G}$ 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

[*]Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden. E-mail: mattias@cs.lth.se, christos@cs.lth.se.

[†]Technical University Eindhoven, Department of Computing Science, P.O. Box 513, 5600 MB Eindhoven E-mail: h.j.gudmundsson@tue.nl

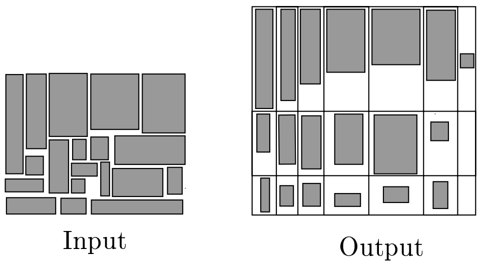

Figure 1: Input is a set of rectangles $\mathcal{S}$. Output a grid packing of $\mathcal{S}$

are 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. More details about the family of grids, called the family of (α, β, γ)*-grids* are given in Section 2 together with two important properties. Then, in Section 3 we show how a grid is tested, and finally, in Section 4, we present the main results.

We will assume that width, height and weight of each rectangle $r \in \mathcal{S}$ is between $[1, n^c]$, for some constant c. We argue in [2] that this assumption can be made without loss of generality with respect to our approximation results.

1 Approximation algorithm

The structure of the approximation algorithm is given below. The two non-trivial steps, lines 10 an 11, will be described in detail in Sections 2 and 3 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.    $bestVal \leftarrow \infty$,
      $\alpha, \beta = \sqrt{1+\varepsilon'}$, $\gamma = \frac{1}{\sqrt{1+\varepsilon'}-1}$
2.    for each $1 \leq i, j \leq \log_\alpha n^c$ do
3.        $\mathcal{S}_{i,j} \leftarrow \emptyset$
4.    for each $r \in \mathcal{S}$ do
5.        $i \leftarrow \lceil \log_\alpha \text{width}(r) \rceil$
6.        $j \leftarrow \lceil \log_\alpha \text{height}(r) \rceil$
7.        $\mathcal{S}_{i,j} \leftarrow \mathcal{S}_{i,j} \cup \{r\}$
8.    end
9.    for $k \leftarrow 1$ to $n^{f(\alpha,\beta,\gamma)}$ do
10.       $\mathcal{G} \leftarrow$ GENERATEGRID($\alpha, \beta, \gamma, k$)
11.       $val \leftarrow$ TESTGRID($\mathcal{G}, \mathcal{S}, \alpha, \beta, \gamma$)
12.       if $val < bestVal$ then
13.           $bestVal \leftarrow val$ and $bestGrid \leftarrow \mathcal{G}$
14.   end
15.   Output PACKINTOGRID($\mathcal{S}, bestGrid$)
```

The initialisation is performed on lines 1 to 3. On lines 4-8, 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 α^{i-1} and α^i, and the height of r is between α^{j-1} and α^j. Lines 1-8 obviously run in linear time. Next, a sequence of grids $\mathcal{G}$ are produced in a loop of lines 9-14. They are the members of the so-called family of (α, β, γ)-grids which is described in Section 2. (It consists of $n^{f(\alpha,\beta,\gamma)}$ grids, where $f(\alpha, \beta, \gamma) = (2c \log(\alpha\beta\gamma))/(\log \alpha \log \beta)$.) 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 the family of (α, β, γ)-grids 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.

2 (α, β, γ)-grids

The aim of this section is to define the family $\mathcal{F}$ of (α, β, γ)-grids and prove two properties about $\mathcal{F}$. 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 there exists a grid $\mathcal{G} \in \mathcal{F}$ that includes G and whose width and height is at most a factor $\left(\frac{\alpha\beta\gamma}{\alpha\gamma-1}\right)$ times larger than the width and height of G, and
2. $\#\mathcal{F} \leq n^{f(\alpha,\beta,\gamma))}$.

The definition of an (α, β, γ)-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.

Consider an arbitrary grid G and let α be a real constant greater than 1. An *α-restricted* grid is a grid where the width and height of each cell in the grid is an integral power of α (multiple of α^i for some integer i).

Let G be a α-restricted grid. If the number of columns/rows of each size is an integral power of β then G is a (α, β)-restricted grid. The columns/rows in an α-restricted grid of width/height α^i are said to have column/row *size i*. A grid G is said to be *γ-monotone* if the number of columns (rows) of size i is at most a factor $\gamma \geq 1$ times smaller than the number of columns (rows) of size $i+1$ for every i.

The following lemma is proven in [2], hence, $\mathcal{F}$ is shown to have Property 1.

Lemma 2 *For any grid G there exists a γ-monotone (α, β)-restricted grid $\mathcal{G}$ (an (α, β, γ)-grid for short) that includes G and whose width and height is at most a factor $\left(\frac{\alpha\beta\gamma}{\alpha\gamma-1}\right)$ 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 $[1..\log_\alpha n^c, 1..\log_\alpha n^c]$ integer matrix, where $\mathcal{G}[i, j]$ stores the number of cells in $\mathcal{G}$ of width α^i and, height α^j. We call this a matrix representation of a grid.

Now we turn our attention to the second property for the family $\mathcal{F}$ of (α, β, γ)-grids, i.e., the number of grids that are members of $\mathcal{F}$ is at most $n^{f(\alpha,\beta,\gamma)}$. Assume that we are given a member $f \in \mathcal{F}$ and that f has c_i columns of size i, $1 \leq i \leq \log_\alpha n^c$, and r_j rows of size j, $1 \leq j \leq \log_\alpha n^c$. Recall that r_j and c_i are integral powers of β. The idea of the scheme is as follows. 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 $\beta^{(\#Rows)}$. Initially $S_{\log_\alpha n^c}$ is the empty string and $\#Rows = 0$. Consider the row size j. We will have two cases, either $r_j \leq (\beta^{\#Rows}/\gamma)$ or $r_j > (\beta^{\#Rows}/\gamma)$. In the first case, add '1' to S_{j+1} to obtain S_j. In the latter case, when $r_j > (\beta^{\#Rows}/\gamma)$, add $(\#Rows - (\log_\beta r_j - \log_\beta \gamma))$ 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 same approach is used to generate the columns, hence we obtain the following observation that proves Property 2.

Observation 3 *S has length $2(\log_\alpha n^c(1 + \log_\beta \gamma) + \log_\beta n)$.*

We obtain the following corollary:

Corollary 4 *Given a bit string B of length $2(\log_\alpha n^c(1 + \log_\beta \gamma) + \log_\beta n)$ one can in time $\mathcal{O}(\log^2 n)$ construct the unique matrix representation of the corresponding (α, β, γ)-grid, or decide that there is no corresponding (α, β, γ)-grid.*

3 Testing a grid

In the previous section we showed a simple method to generate all possible (α, β, γ)-restricted grids. For the approximation algorithm, shown in Section 1, 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 (α, β, γ)-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 α^p and height α^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.

Level 1. Contains the source node $\nu^{(1)}$.

Level 2. Contains $\log_\alpha^2 n^c$ 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 belongs to $\mathcal{S}_{i,j}$.
Level 3. Also contains $\log_\alpha^2 n^c$ 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 $C_{p,q}$. All the edges from level 2 to level 3 have capacity n.
Level 4. Contains 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 capacity equal to the number of cells in $\mathcal{G}$ that belongs to $\mathcal{C}_{p,q}$.

The following results are 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.*

In 1998 Goldberg and Rao [4] 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 followed by the first grid packing theorem.

Lemma 6 *Given a matrix representation of an (α,β,γ)-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 $\mathcal{O}(\log^9 n)$. (2) An optimal packing can be computed in time $\mathcal{O}(\log^9 n + k)$, where k is the number of rectangles in a grid packing of $\mathcal{S}$ in $\mathcal{G}$.*

Theorem 7 *Given a set of rectangles $\mathcal{S}$ and an $\epsilon > 0$, algorithm* GridPack *produces a grid packing whose area is most $(1+\varepsilon)$ times larger than a minimum area grid packing of $\mathcal{S}$ in time $\mathcal{O}(n^{f(\sqrt{1+\varepsilon})}\log^{19/3} n + n)$, where $f(\chi) = \frac{2c\log(\chi/(\sqrt{\chi}-1))}{\log^2\chi})$.*

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 ε appropriately we obtain that, algorithm GridPack is a PTAS for the MAGP-problem, and if ε is set to be a large constant GridPack produces a grid packing that is within a constant factor of the optimal in linear time. Note also that the approximation algorithm easily can be generalised to d dimensions.

4 Results

The approximation algorithm presented above can be extended and generalised to variants of the basic grid packing problem by performing some small modifications to the procedure TestGrid. We obtain the following corollary.

Corollary 8 *Algorithm* GridPack *is a τ-approximation algorithm with time complexity $\mathcal{O}(n^{f(\sqrt{1+\varepsilon})}\log^{19/3} n + n)$, where:*
- *$\tau = (1+\varepsilon)$ for the problems MAkGP, MAGPAR and MDGP,*
- *$\tau = (1-\varepsilon)$ for the MWP-problem, and*
- *$\tau = ((\lfloor 2(1+\varepsilon)\rfloor)^2)$ for the MNWP-problem*

Finally we consider the hardness of three variants of the MAGP-problem.

Theorem 9 *(1) MNWP cannot be approximated within a factor of $3/2-\varepsilon$ for any $\varepsilon > 0$, unless $P = NP$. (2) The MAGPAR-problem and the MWP-problem are NP-hard.*

5 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" [1, 3, 5, 6].

References

[1] R. J. Anderson and S. Sobti. The Table Layout Problem. Proc. of SoCG, 1999.

[2] M. Andersson, J. Gudmundsson and C. Levcopoulos. Tech rep., Dept. of Comp. Sci., Lund University, 2002.

[3] R.J. Beach. Setting tables and illustrations with style. PhD thesis, Dept. of computer science, University of Waterloo, Canada, 1985.

[4] A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. Journal of the ACM, 45(5):783–797, 1998.

[5] K. Shin, K. Kobayashi and A. Suzuki. TAFEL MUSIK, formatting algorithm of tables. Proc. of Principles of Document Processing, 1994.

[6] X. Wang and D. Wood. Tabular formatting problems. Proc. of Principles of Document Processing, 1996.