SIMPLE ALGORITHMS FOR RECTANGLE PACKING PROBLEM – FROM THEORY TO EXPERIMENTS

Adam KURPISZ

Abstract: In this paper we consider a Rectangle Packing Problem which is a part of production process in many branches of industry, i.e. VLSI circuit design, stone-cutting etc. We consider a slightly modified Rectangle Packing Problem where for each rectangle (module) its width and height come from uniform probability distribution between 0 and 1 - U(0,1). We provide a family of simple algorithms, together with a theoretical analysis and computer simulations to prove their effectiveness. All the algorithms run in polynomial time and provide a near optimal solution.

Keywords: production engineering; optimization; Rectangle Packing Problem

1. Introduction, definitions, notation

Nowadays, due to the cost optimization, planning a production process is a challenging task. In many branches of industry one can face a problem of minimizing the waste of material used for producing some elements. One of such problems is a Rectangle Packing Problem (RPP). The RPP is a discrete optimization problem [3], which could be solved by variety of methods developed in the past four decades [2]. We start with a formal definition of a Rectangle Packing Problem [1]. A set $B = \{1, \dots, n\}$ of n rectangles is given. Each rectangle $i \in B$ is characterized by its width x_i, height y_i and area $a_i = x_i \cdot y_i$. The rotation (by 90°) of the rectangles is allowed. The goal of the Rectangle Packing Problem (RPP) is to find such packing, i.e., placement coordinates of a bottom-left corner of each block $i \in B$, that any two blocks do not overlap and the area of minimum enclosing rectangle of the packing is minimized. We denote as b_x, b_y the width and height of the enclosing rectangle, respectively. As the measure of the solution we use $VoFP$ (Value of Filling Percentage).

$$VoFP = \frac{\sum_{i \in B} x_i \cdot y_i}{b_x \cdot b_y} \tag{1}$$

In this paper we consider modification of above problem. Let X_i, Y_i for $i \in B$ be the independent random variables from continuous uniform distribution $U(0,1)$ where 0 and 1 are the minimum and the maximum achievable values, respectively. The values of X_i, Y_i describes the width and height of the i'th rectangle. We consider continuous uniform distribution, thus all the values from $[0,1]$ are equiprobable. Now, in a similar way B_x, B_y are some random variables which values describe the width and height of the enclosing rectangle. We consider the expected value of $VoFP$ as a measure of effectiveness of an algorithm.

$$E[VoFP] = \frac{E[\sum_{i \in B} X_i \cdot Y_i]}{E[B_x \cdot B_y]} \tag{2}$$

2. Analysis of the algorithms

In this section we provide a family of simple algorithms for Rectangle Packing Problem together with some analysis of their effectiveness.

2.1. Algorithm 1

In this section we consider a very simple algorithm and we provide a probabilistic analysis of its solution.

```
    Algorithm 1
1.  sort rectangles with respect to their $y_i$ value
2.  **from** 1 **to** n **do:**
3.      take $i$'th rectangle and put inline close to the previous $i-1$ rectangles
4.  calculate the width $b_x$ and height $b_y$ of enclosing rectangle
5.  **return** $b_x \cdot b_y$
```

An example of resulting packing of Algorithm 1 can be seen in Figure 1.

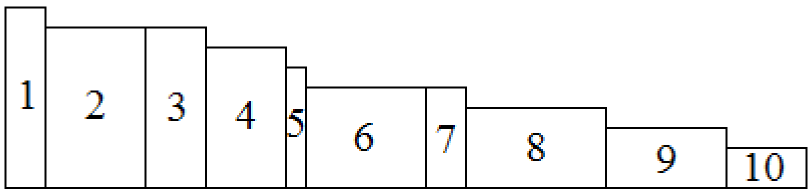

Figure 1 Example of packing obtained with Algorithm 1

We start the analysis with recalling the probability density function (PDF) and cumulative density function (CDF) for continuous uniform distribution $U(0,1)$ denoted as $f(x)$ and $F(x)$ respectively:

$$f(x) = 1, \tag{3}$$
$$F(x) = x\,. \tag{4}$$

Since we sort all the rectangles with respect to their height we have

$$b_y = max\{y_1, y_2, \dots, y_n\}. \tag{5}$$

Thus we calculate the probability density function of the highest rectangle over all n given rectangles. The PDF of the highest rectangle can be expressed as the n order statistic from the set of n random variables. The general formula of PDF for k ordered statistic can be formed as

$$f_{k:n}(x) = n\binom{n}{k} f(x)F(x)^{k-1}(1-F(x))^{n-k}. \tag{6}$$

Since we consider the continuous uniform distribution $U(0,1)$ and we look for the maximum height over n rectangles, thus we look for the n ordered statistic, which takes the form

$$B_y \sim f_{n:n}(x) = nf(x)F(x)^{n-1} = nx^{n-1}. \tag{7}$$

Now we calculate the expected value for of a random variable B_y

$$E[B_y] = \int_0^1 nx^n dx = \frac{n}{n+1}. \tag{8}$$

Since we do not interfere in the values of x_i, and random variables X_i are independent, the expected value of width of enclosing rectangle ($E[B_x]$) can be formed as

$$E[B_x] = E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i] = \frac{n}{2}. \tag{9}$$

The last thing is to calculate the expected value of an area of a particular rectangle. Since for each $i \in B$, X_i and Y_i are independent, then

$$E[X_i \cdot Y_i] = E[X_i] \cdot E[Y_i] = \frac{1}{4}. \tag{10}$$

Thus finally, the expected value of *VoFP* is equal to

$$E[VoFP] = \frac{\frac{n}{4}}{\frac{n}{n+1} \cdot \frac{n}{2}} = \frac{n+1}{2n} \tag{11}$$

and with $n \to \infty$ the filling percentage is close to $1/2$. To compare the theoretical results with computer simulations go to the Section 3.

2.2. Algorithm 2

The next algorithm we take into consideration is a small modification of the algorithm presented in the previous subsection. This time we also make an inline packing but now before the rectangles are scheduled, we rotate each rectangle to ensure that its height exceeds its width. Thus we decrease the potential total width of enclosing rectangle b_x with only small increasing of the total height b_y. The formal description of an algorithm is as follow (Algorithm 2):

```
    Algorithm 2
1.  **from** 1 **to** n **do:**
2.    **if** width of the $i$'th rectangle is bigger than its height
3.      rotate the rectangle
4.  sort rectangles with respect to their $y_i$ value
5.  **from** 1 **to** n **do:**
6.    take $i$'th rectangle and put inline close to the previous $i$-$1$ rectangles
7.  calculate the width $b_x$ and height $b_y$ of enclosing rectangle
8.  **return** $b_x \cdot b_y$
```

An example of resulting packing of Algorithm 2 can be seen in Figure 2.

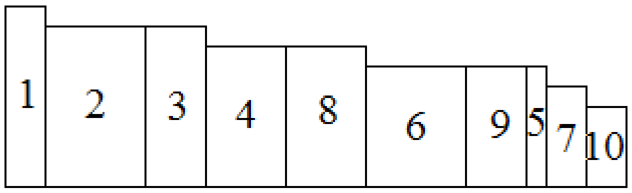

Figure 2 Example of packing obtained with Algorithm 2

Once again the key observation is that we have to calculate the expected value of the height $(E[B_y])$ and the width $(E[B_x])$ of the enclosing rectangle. We start with the $(E[B_y])$ value. Since we allow rotating the rectangles, the maximum height over n rectangles is indeed the maximum value over all original (before rotating) values of x_i and y_i:

$$b_y = max\{x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n\}. \tag{12}$$

The PDF of the random variable B_y can now be formed as the maximum over $2n$ uniform random variables:

$$B_y \sim f_{2n:2n}(x) = 2nf(x)F(x)^{2n-1} = 2nx^{2n-1} \tag{13}$$

and thus

$$E[B_y] = \int_0^1 2nx^{2n}dx = \frac{2n}{2n+1}. \tag{14}$$

Calculating the expected value of B_x is now more involving than in the previous subsection. Every single rectangle's width does not come from the uniform distribution, but from the minimum over its height and width, thus

$$X_i \sim f_{1:2}(x) = 2f(x)\big(1 - F(x)\big) = 2(1 - x) \tag{15}$$

and using similar formula as in (8) one can calculate the expected value of X_i:

$$E[X_i] = \int_0^1 x2(1-x) = \frac{1}{3}. \tag{16}$$

The expected value of width of enclosing rectangle ($E[B_x]$) can be formed as

$$E[B_x] = E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i] = \frac{n}{3}. \tag{17}$$

Now using (10), the expected value of *VoFP* is equal to

$$E[VoFP] = \frac{\frac{n}{4}}{\frac{2n}{2n+1}\cdot\frac{n}{3}} = \frac{2n+1}{2n}\cdot\frac{3}{4} \tag{18}$$

and, again, with $n \to \infty$ the filling percentage is close to $^3/_4$. With comparison to the previous algorithm we increase the *VoFP* with 25% points. Simulation confirming above calculations is available in Section 3.
In next two sections we describe a modification of Algorithm 1 and 2, respectively.

2.3. Algorithm 3

The main modification of an algorithm considered in this section is to allow the schedule to be multilevel. We put one rectangle next to the other and at particular point we start the next level, i.e. we put the rectangle on the top of the first rectangle put in the schedule. The number of levels we choose as $\sqrt{n}$. In this subsection (similar to the subsection 1.1) we do not allow the rotation. The formal description of the algorithm can be seen below.

```
    Algorithm 3
1.  sort rectangles with respect to their $y_i$ value
2.  **from** 1 **to** $\sqrt{n}$ **do:**
3.    **from** 1 **to** $\sqrt{n}$ **do:**
4.        take $i$'th rectangle and put inline close to the previous rectangle
5.    start a new level
6.  calculate the width $b_x$ and height $b_y$ of enclosing rectangle
7.  **return** $b_x \cdot b_y$
```

An example of resulting packing of Algorithm 3 can be seen in Figure 3.
We do not provide an exact calculation on the effectiveness of the considered algorithm, the analysis is much more sophisticated. Instead we provide a computer simulation. All the results of the simulation are available in the Section 3, together with the comparison to the previous algorithms.

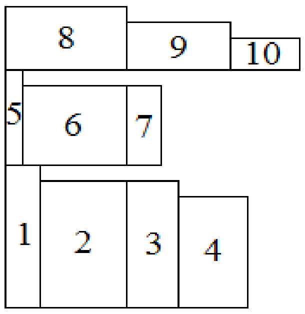

Figure 3 Example of packing obtained with Algorithm 3

2.4. Algorithm 4

In this section we present the multilevel analog of the Algorithm 2. At the very beginning we allow the rotations (to make the rectangles higher than wider) and while the rectangles are scheduled one by one, at particular points we start a new layer of rectangles. Since we prefer the enclosing rectangle to be square shape, and because of the rotating step we decrease the number of layers to the $\frac{2\sqrt{n}}{3}$. The formal description is as follow:

```
Algorithm 4
1.  from 1 to n do:
2.    if width of the $i$'th rectangle is bigger than its height
3.      rotate the rectangle
4.  sort rectangles with respect to their $y_i$ value
5.  from 1 to $\frac{2\sqrt{n}}{3}$ do:
6.    from 1 to $\frac{3\sqrt{n}}{2}$ do:
7.      take $i$'th rectangle and put inline close to the previous rectangle
8.    start a new level
9.  calculate the width $b_x$ and height $b_y$ of enclosing rectangle
10. return $b_x \cdot b_y$
```

An example of resulting packing of Algorithm 4 can be seen in Figure 4.

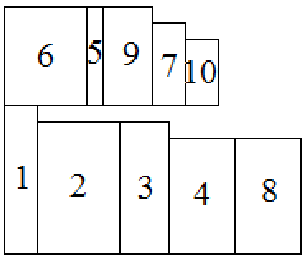

Figure 4 Example of packing obtained with Algorithm 4

As in the previous subsection we skip the probabilistic analysis and make a computer simulation instead. For detailed information see Section 3.

3. Computer simulation

In this section we provide a complete computer simulation for all considered algorithms. For Algorithm 1 and 2 we compare the results calculated in Sections 2.1 and 2.2 with the one obtained with computer simulation. For Algorithms 3 and 4 we present the simulated $VoFP$. Table 1 provides a complete comparison for an instance composed of 10000 rectangles.

Table 6 Expected value of VoFP for Algorithms 1-4

	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4
Theoretic $E[VoFP]$	0,5	0,75	-	-
Simulated $E[VoFP]$	0,503298	0,750491	0,978394	0,981380

4. Conclusions

In this paper we analyze four different algorithms for Rectangle Packing Problem with data random from continuous uniform distribution $U(0,1)$. We show that even a simple procedure can lead to very efficient solution. For Algorithm 3 and 4 the $VoPF$ equals to 0,98. On the other hand we show that unless the problem is considered in full generality (with arbitrary data), using more sophisticated methods for Rectangle Packing Problem with random data is useless since a simple procedure guarantees a very good result.

Acknowledgements
Adam Kurpisz work is co-financed by the European Union as part of the European Social Fund.

Bibliography

1. Baker B. S., Coifman E. G., Rivest R. L. Orthogonal packings in two dimensions. SIAM J. Comput., 9, (4), 1980, 846–855.
2. Blum C., Roli A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys, 35, (3), 2003, 268-308.
3. Cormen T., Leiserson C., Rivest R. Introduction to algorithms. McGraw-Hill Book Company, 1990.

Mgr inż. Adam Kurpisz
Instytut Matematyki i Informatyki
Politechnika Wrocławska
50-372 Wrocław, ul. Janiszewskiego 14a
tel./fax: (0-71) 320-30-29
e-mail: adam.kurpisz@pwr.wroc.pl