OTTO-VON-GUERICKE-UNIVERSITY MAGDEBURG
FACULTY OF ECONOMICS AND MANAGEMENT

SLOPPGEN: A Problem Generator for the Two-Dimensional Rectangular Single Large Object Placement Problem With a Single Defect

Vera Neidlein • Gerhard Wäscher

FEMM Working Paper No. 15, July 2008

F E M M

Faculty of Economics and Management Magdeburg

Working Paper Series

Otto-von-Guericke-University Magdeburg
Faculty of Economics and Management
P.O. Box 4120
39016 Magdeburg, Germany
http://www.ww.uni-magdeburg.de/

Vera Neidlein • Gerhard Wäscher

SLOPPGEN:
A Problem Generator for the Two-Dimensional Rectangular Single Large Object Placement Problem With a Single Defect

July 2008

Abstract: In this paper, a problem generator for the Two-Dimensional Rectangular Single Large Object Placement Problem is presented. The parameters defining this problem are identified and described. The features of the problem generator are pointed out, and it is shown how the program can be used for the generation of reproducible random problem instances.

Keywords: two-dimensional cutting, defect, problem generator

Corresponding author:

Dipl.-Math. oec. Vera Neidlein

Otto-von-Guericke-University Magdeburg
Faculty of Economics and Management
- Management Science -
Postfach 4120
39016 Magdeburg
{vera.neidlein@ww.uni-magdeburg.de}

Table of Contents

1 Introduction

Two-dimensional cutting problems have been in the focus of researchers for many years. Since its beginnings (cf. Brooks et al. (1940), Gilmore and Gomory (1965)), different types of this problem have been widely studied in the literature. Starting in the 1960s, work has also been done on problems including material defects (e.g. Hahn (1968), Carnieri et al. (1993)), which are particularly relevant in practice.

There are still numerous publications in the field of two-dimensional cutting at the present time. Many of them deal with the challenge of developing exact algorithms that are faster than existing ones, or with developing faster and better-performing heuristics. The latter is due to the fact that most problems in the field of two-dimensional cutting are known to be NP-hard.

It is often difficult to draw general conclusions on the performance of an algorithm, because authors – in particular when dealing with real-world, "practical" problems – often only give the results for a few instances to demonstrate how their algorithm works. These instances usually either come from the practical problem considered, or they are taken from the literature (e.g. the OR library by Beasley (1990), problems presented by Herz (1972), Beasley (1985)) and treated as benchmark problems, although – sometimes having been used for decades – it is not known whether they still can be seen as valid benchmark problems. In both cases, it is almost impossible to transfer the developed insights to problem instances with different data structures. Thus, systematical tests gain more importance with every new publication in this field due to the huge and still growing number of algorithms for cutting and packing problems.

In order to overcome this lack of appropriate sets of test problem instances and provide general access to an un-biased basis of problem data for an important class of cutting problems, namely the so-called Two-Dimensional Rectangular Single Large Object Placement Problem (cf. Wäscher et al. (2007)), a problem generator has been developed that will be described in this publication. It can be used to easily generate a large number of problem instances with specific desired properties, and can thus enable researchers to follow a more systematic approach for testing algorithms with regard to the performance compared to existing algorithms and with regard to instances from problem classes of different data structures.

This paper is organized as follows: In Section 2, we introduce the Two-Dimensional Rectangular Single Large Object Placement Problem with a single defect, for which this problem generator was designed. In Section 3, the parameters of the problem are identified and explained, which can be used to define classes of problems. Hence, homogeneous instances of a specific problem class can be generated randomly with the problem generator described in detail in Section 4. In Section 5, examples for instances from a specific problem class are given.

2 The Two-Dimensional Single Large Object Placement Problem

2.1 Standard Problem

The Two-Dimensional Rectangular Single Large Object Placement Problem (2D_R_SLOPP) is a standard Cutting&Packing problem which can be described as follows (cf. Wäscher et al. (2007)):

A large rectangle (large object) of given length L and width W is to be cut down in order to provide smaller rectangles (small items) of particular dimensions (types), i.e. of length l_i, width w_i and value v_i (i = 1,…,n). It is imposed that all cuts lie parallel to one of the edges of the large object (orthogonal layout). The objective is to maximize the total value of the provided small items (output maximization problem). In our special case, the value of the item types corresponds to their area, i.e. $v_i = l_i \cdot w_i$ (unweighted problem).

As a first-level standard problem (see Wäscher et al. (2007)), the 2D_R_SLOPP is characterized by the absence of additional constraints, i.e. there are no upper or lower bounds on the number of times an item type has to be cut from the large object (unconstrained problem), the items may have any orientation (vertical or horizontal) on the large object (no rotational constraint), the type of cutting is not restricted (non-guillotineable-constrained layout), and there are no upper or lower bounds on the number of cutting stages (non-staged problem).

Any layout of an assortment of small items on the large object such that the items do not overlap and lie entirely within the large object is called a feasible cutting pattern.

The 2D_R_SLOPP can now be formulated as a mathematical model:

$$\max \quad \sum_{i=1}^{n} v_i \cdot z_i$$

s.t. $(z_1,...,z_n)$ corresponds to a feasible cutting pattern.

Herein, (z_1,…,z_n) is said to correspond to a feasible cutting pattern if item type i (i = 1, …, n) can be laid out z_i times an on the large object in a way that a feasible cutting pattern is obtained.

All the problem data introduced above can be assumed to be integer numbers without loss of generality since – as similarly described in Gau and Wäscher (1995) for a one-dimensional problem – the data L, W, l_1,…,l_n, w_1,…,w_n is only implicitly represented in the model formulation as it defines the feasible cutting patterns. Therefore, multiplying all the data with a constant factor $m > 0$ results in an identical model formulation.

2.2 Problem Variant: Single Defect

The standard problem can be extended by introducing a defect on the large object. This defect is defined by its length l_d and width w_d and by the position of its lower left corner (x_d,y_d). In other words, due to some material deficiency, there is a region on the large object to which no small item is to be assigned. This region is contained in the smallest possible rectangle with edges parallel to the edges of the large object. Figure 1 shows an example for such a defect. In every feasible cutting pattern, no small item must overlap with this rectangle.

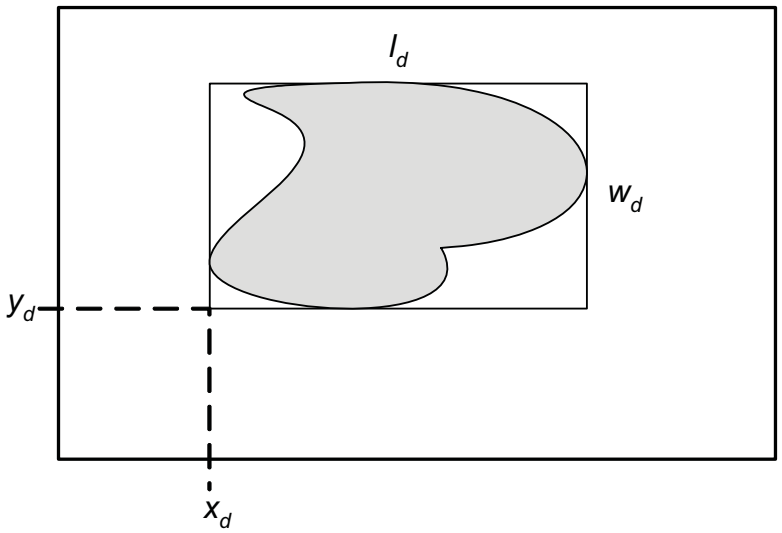

Fig. 1: *Representation of a single defect on the large object*

The above-given mathematical formulation remains valid for this problem variant, since the defect is only implicitly contained in the model via the postulation of a feasible cutting pattern.

3 Identification of Problem Parameters

In this section, the parameters of the 2D_R_SLOPP with a single defect (2D_R_SLOPP_1DEF) will be identified, and it will be explained why they are a sensible choice and how they will be included in the problem generator. These parameters (or, more precisely, a specific set of parameter values) are then used as descriptors for homogeneous problem classes.

Table 1 gives an overview of the parameters of the 2D_R_SLOPP_1DEF, which will be described in more detail in the following sections.

n	number of small item types
(L,W)	dimensions of the large object
s	relative size of the item types w.r.t. the size of the large object
(l_d,w_d)	dimensions of the defect
(x_d,y_d)	location of the defect

Table 1: *Problem parameters of the 2D_R_SLOPP_1DEF*

3.1 Number of Small Item Types

The number n of small item types is a very straightforward candidate for a parameter of the 2D_R_SLOPP_1DEF, as it is a common measure for the problem size. It can be expected that the problem becomes more difficult to solve with a growing number of item types as the number of possible cutting patterns increases (for example, for only one item type, the optimal solution can be calculated applying a simple formula,

at least if a guillotineable layout is assumed), yet it is expected that the solution quality also increases.

In the problem generator, this parameter is introduced as a controllable one, i.e. it can be fixed to an appropriate value by the user.

3.2 Dimensions of the Large Object

It has already been said in Section 2 that the problem data can be multiplied by a constant factor without changing the model. Hence, the absolute size of the large object is not crucial, but the relative size in relation to the small item types and the defect. Yet, it can be assumed that the shape of the large object (quadratic, rectangular) is an influencing factor on how difficult it is to solve the problem, as a quadratic large object is probably easier to cut down than a very long and very narrow large object of the same area. As a consequence, we introduce the length and width of the large object as problem parameters.

Again, these parameters are controllable and can be explicitly defined by the user of the problem generator.

3.3 Relative Size of the Item Types

The sizes of the small item types are also likely to have an impact on the difficulty of the 2D_R_SLOPP_1DEF. We measure the size of an item type by its area which is given as $a_i = l_i \cdot w_i$. Small items allow for more and more sophisticated cutting patterns so that the solution quality is expected to increase, whereas the solution time is expected to drop.

The influencing factor is not the absolute, but the relative size of the item types in relation to the area of the large object. This relative size can be controlled by the user via the two descriptors $\underline{s}$ and $\overline{s}$, which represent the lower and the upper bound for the relative size of the item types w.r.t. the area of the large object.

As it seems not reasonable – and not manageable – to keep control over every possible shape of the item types, the area of the item types is a sensible-chosen representative for the size. It will be described in more detail in Section 4 how the generation of “degenerated” item types (very long but very narrow) can be avoided.

3.4 Dimensions of the Defect

As we consider a cutting problem including a material defect, at least one parameter has to be related to the defect. One candidate for this is its size. Unlike the small item types, length and width of the defect rather than its area are identified as influencing factors and as controllable parameters because, as there is only a single defect present, the shape of the defect can be assumed to have a strong influence on how difficult a problem is to solve. Although having the same area, a narrow defect which almost divides the large object in two and thus practically leaves us with two standard 2D_R_SLOPP is expected to form a very different challenge than a big quadratic defect on the large object.

3.5 Location of the Defect

The location of the defect on the large object (represented by its lower left corner) is not included as a controllable problem parameter, but is depicted as a realization of a random variable. The reason for this is that its influence on the difficulty of the problem can be assumed to be strongly dependent on the size and shape of the large object and the defect and can thus not be defined independently in a sensible way.

4 Generation of Test Problem Instances

In the course of testing algorithms, a random sample of problem instances from a set of problem classes of the 2D_R_SLOPP_1DEF will have to be provided. Each problem class is defined by the set of parameters (or descriptors) (L, W, n, $\underline{s}, \overline{s}$, l_d, w_d) specified above.

A specific problem class given through the values of L, W, n, $\underline{s}$, $\overline{s}$, l_d, and w_d forms the basis for problem instances which can be interpreted as a realization of a random variable ($l_1,\ldots,l_n$, $w_1,\ldots,w_n$, x_d,y_d). For the generation of a problem instance, these values have to be fixed randomly. Section 4.2 deals with generating the dimensions of the item types, whereas in Section 4.3, the generation of the location of the defect is considered in detail. Before that, a short description of the pseudo-random number generator used within this implementation is given.

4.1 Generation of Uniformly Distributed Pseudo-Random Numbers

Instead of using any built-in pseudo-random number generator (which may be dependent on the computer the program is running on), the problem generator includes an implementation, which ensures the portability of the generator as well as the reproducibility of the results. The implemented pseudo-random number generator is the one that has also been used by Gau and Wäscher (1995), and which is a special variant of the method attributed to Lehmer (Hutchinson (1966)). Only a very brief description shall be given here, similar to the one in Gau and Wäscher (1995).

Let a prime number p and an integer $c \in \{1,\ldots,p\text{-}1\}$ be given. After defining an integer number $r_1 \in \{1,\ldots,p\text{-}1\}$ as a "seed" (an initial number), a sequence of uniformly distributed integer random numbers is defined through the recursive formula

$$r_{n+1} = c \cdot r_n \bmod p\,.$$

A sequence of uniformly distributed random numbers in the interval (0,1) can thus be obtained by dividing the integer random numbers by p.

An appropriate choice for p and c (Park and Miller (1988)) is

$$p = 2^{31} - 1 = 2{,}147{,}483{,}647 \quad \text{and} \quad c = 16{,}807\,.$$

Due to the size of the numbers, a direct calculation of the product of c and r_n is not possible on most contemporary computers. Yet, this problem can be solved by an approximate factorization of p in the following way:

Let

$$p = c \cdot q + t$$

where

$q = p$ div c (integer part of p) and $t = p \bmod c$.

With values for p and c as introduced above, we obtain

$q = 127{,}773$ and $t = 2{,}836$.

The desired value $c \cdot r_n \bmod p$ can now be calculated as follows (involving only numbers up to p):

$$c \cdot r_n \bmod p = \begin{cases} c \cdot (r_n \bmod q) - t \cdot (r_n \operatorname{div} q) & \text{if } c \cdot (r_n \bmod q) - t \cdot (r_n \operatorname{div} q) \geq 0 \\ c \cdot (r_n \bmod q) - t \cdot (r_n \operatorname{div} q) + p & \text{otherwise.} \end{cases}$$

The seed r_1 has not been chosen in advance. Thus, if this problem generator is used for testing algorithms, the seed used should always be published together with the results, due to reproducibility of the test data.

4.2 Determination of Item Type Dimensions

It has already been stated that all dimensions (l_i, w_i), $i = 1,\ldots,n$, of the item types can be considered as integer values without loss of generality. The following procedure is used to determine the values: A realization $\hat{a}_i$ (the preliminary area of item type i) of a random variable A_i which is uniformly distributed in the interval $[\underline{s} \cdot LW, \overline{s} \cdot LW]$ is generated. Note that this number is not required to be integer. To avoid the generation of biased items, which would occur if length and width would be generated directly, we generate the aspect ratio

$$b_i = \frac{l_i}{l_i + w_i} \tag{1}$$

and use this value to calculate l_i and w_i.

To guarantee non-biased problem instances, every item type generated fits on the large object. Therefore it is necessary to limit b_i to an interval chosen in a way that l_i will not exceed L and w_i will not exceed W. In detail, the following conditions hold if item type i fits on the large object:

$$l_i \cdot w_i = \hat{a}_i \tag{2}$$

$$l_i \leq L \tag{3}$$

$$w_i \leq W \quad \Leftrightarrow \quad \frac{\hat{a}_i}{l_i} \leq W \quad \Leftrightarrow \quad \frac{\hat{a}_i}{W} \leq l_i \tag{4}$$

(2) can be written as $w_i = \frac{\hat{a}_i}{l_i}$. Inserting this into (1) and solving for l_i gives

$$l_i = \sqrt{\frac{\hat{a}_i \cdot b_i}{1 - b_i}}. \tag{5}$$

Inserting this formula for l_i into (3) and (4) gives the following feasible interval for b_i:

$$\frac{\hat{a}_i}{\hat{a}_i + W^2} \leq b_i \leq \frac{L^2}{\hat{a}_i + L^2}. \tag{6}$$

Furthermore, it is desirable to avoid “degenerated” item types which are very long, but very narrow. Thus b_i is limited to the interval $[0.1, 0.9]$. If necessary and desired, the interval can be adjusted in the source code.

Summing up, b_i is generated as a realization of a random variable B_i which is uniformly distributed in the interval $[\underline{b_i}, \overline{b_i}]$ where

$$\underline{b_i} = \max\left\{\frac{\hat{a}_i}{\hat{a}_i + W^2}, 0.1\right\} \text{ and} \tag{7}$$

$$\overline{b_i} = \min\left\{\frac{L^2}{\hat{a}_i + L^2}, 0.9\right\}. \tag{8}$$

Preliminary values for l_i and w_i are then calculated as follows:

$$\hat{l}_i = \sqrt{\frac{\hat{a}_i \cdot b_i}{1 - b_i}} \quad \text{and} \tag{9}$$

$$\hat{w}_i = \frac{\hat{a}_i}{\hat{l}_i}. \tag{10}$$

Both values are rounded to the nearest integer to obtain l_i and w_i, which gives an item type with an area $a_i = l_i \cdot w_i$ which is very close to the randomly generated value $\hat{a}_i$.

Due to the rounding throughout the process, it may well happen that the area a_i does not lie within the feasible interval $[\underline{s} \cdot LW, \overline{s} \cdot LW]$. If this is the case, the following adjustment procedure is applied. Assuming that a_i is too small, item type i is then adapted in the following way (an analogous procedure is used if a_i is too large): A realization k of a random variable K which is uniformly distributed in the interval [0,1] is generated. If $k < 0.5$ and if $l_i < L$, l_i is augmented by 1. Otherwise w_i is augmented by 1, except if $w_i = W$, then we are back to augmenting l_i. (Note that $l_i = L$ and $w_i = W$ are mutually exclusive for all reasonable parameter choices.) The area a_i is recalculated, and it is checked if it lies within the feasible interval now. If this is not the case, the adjustment procedure starts again.

4.3 Determination of the Location of the Defect

The position of the lower left corner of the defect, (x_d, y_d), has to fulfill the following properties: x_d must lie within the interval $[0, L - l_d]$ and y_d must lie within the interval $[0, W - w_d]$. Thus, realizations $\hat{x}_d$ of a random variable X, which is uniformly distributed in the interval $[0, L - l_d]$, and $\hat{y}_d$ of a random variable Y, which is uniformly distributed in the interval $[0, W - w_d]$ are generated. These values are then rounded mathematically to obtain x_d and y_d.

4.4 Implementation

The problem generator described in this section has been coded in C and compiled with the free compiler Bloodshed Dev-C++ (in order to enable everyone interested to recompile the code with adapted specifications). The problem parameters can be entered via a dialog window when running the program.

The implementation includes a special feature regarding the defect. For understanding the performance of an algorithm, it can be useful to solve the same instance (item types) with several different defects of, for example, different sizes. To reduce the effort for the user, the problem generator can generate the position of more than one defect for each problem instance; the sizes of these defects are given to the program as parameters. Yet, each defect has to be treated as an individual, single defect for the particular set of item types it is generated for, as it is not checked if the defects overlap.

Note again that, as has been said before, the problem generator uses the area of each item type as its value rather than generating a random value.

A version executable under Windows as well as the source code of the problem generator are available at www.ovgu.de/mansci/materials.

5 Numerical Example

To demonstrate the functioning of the problem generator, a few examples of instances from a specific problem class will be shown in this section. In particular, a special feature of the problem generator regarding the randomly generated defect will be explained.

In order to initialize the pseudo-random number generator, the value 123456 has been used as a seed. For the examples presented in this section, the parameters of the problem have been chosen as given in Table 2.

number of instances	50
dimensions of the large object	(400,300)
number of item types	10
lower bound for relative size of item types	0.01
upper bound for relative size of item types	0.05
number of defects per instance	3
dimensions of the 1st defect	(5,5)
dimensions of the 2nd defect	(10,15)
dimensions of the 3rd defect	(40,20)

Table 2: *Parameter choices for the example*

From the 50 problem instances provided by the problem generator, the first and the last one are given in Table 3.

Instance 01					Instance 50				
item types					item types				
i	length	width	area		i	length	width	area	
1	15	122	1830		1	40	87	3480	
2	31	57	1767		2	43	92	3956	
3	172	30	5160		3	94	46	4324	
4	46	122	5612		4	72	57	4104	
5	72	47	3384		5	208	24	4992	
6	51	30	1530		6	19	138	2622	
7	23	79	1817		7	50	48	2400	
8	58	76	4408		8	15	118	1770	
9	143	21	3003		9	16	94	1504	
10	21	75	1575		10	53	77	4081	
defects					defects				
no.	lower left corner		upper right corner		no.	lower left corner		upper right corner	
1	232	102	237	107	1	289	242	294	247
2	58	243	68	258	2	364	49	374	64
3	191	11	231	31	3	190	204	230	224

Table 3: *Selected problem instances of the numerical example*

6 Remarks

The problem generator described in this paper has been developed in order to be able to test algorithms for the 2D_R_SLOPP_1DEF. By ignoring the data of the defect, the problem instances can also be applied to testing algorithms for the 2D_R_SLOPP without any defect. Apart from that, it is also directly applicable for both the 2D_R_SLOPP_1DEF and the 2D_R_SLOPP if the rotation constraint is imposed and / or if a guillotineable cutting pattern layout is required.

The problem instances generated are suitable for unweighted problems (the value of each item type equals its area). Only minor modifications in the code are necessary to enable the problem generator to provide a specific value for each item type and, thus, instances for weighted problems.

References

Beasley, J.E. (1985):
Algorithms for Unconstrained Two-Dimensional Guillotine Cutting. *Journal of the Operational Research Society* **36**, 297-305.

Beasley, J.E. (1990):
OR-Library: Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society* **41**, 1069-1072.
Library available online at http://people.brunel.ac.uk/~mastjjb/jeb/info.html, last visited 2008/06/18

Brooks, R.L.; Smith, C.A.B.; Stone, A.H., Tutte, W.T. (1940):
The Dissection of Rectangles into Squares. *Duke Mathematical Journal* **7**, 312-340.

Carnieri, C.; Mendoza, G.A.; Luppold, W.G. (1993):
Optimal Cutting of Dimension Parts from Lumber with a Defect. *Forest Products Journal* **43**, 66-72.

Christofides, N.; Whitlock, C. (1977):
An Algorithm for Two-Dimensional Cutting Problems. *Operations Research* **25**, 30-44.

Gau, T.; Wäscher, G. (1995):
CUTGEN1: A Problem Generator for the Standard One-Dimensional Cutting Stock Problem. *European Journal of Operational Research* **84**, 572-579.

Gilmore, P.C.; Gomory, R.E. (1965):
Multistage Cutting Stock Problems of Two and More Dimensions. *Operations Research* **13**, 94-120.

Herz, J.C. (1972):
Recursive Computational Procedure for Two-Dimensional Stock Cutting. *IBM Journal of Research and Development* **16**, 462-469.

Hutchinson, D.W. (1966):
A New Uniform Pseudorandom Number Generator. *Communications of the ACM* **9**, 432-433.

Park, S.K.; Miller, K.W. (1988):
Random Number Generators: Good Ones are Hard to Find. *Communications of the ACM* **31**, 1192-1201.

Wäscher, G.; Haußner, H.; Schumann, H. (2007):
An Improved Typology of Cutting and Packing Problems. *European Journal of Operational Research* **183**, 1109-1130.