Volume 31 (2), pp. 77–94
http://orion.journals.ac.za

ORiON
ISSN 0259–191X (print)
ISSN 2224–0004 (online)

An exact algorithm for the N-sheet two dimensional single stock-size cutting stock problem

T Steyn* JM Hattingh†

Received: 3 December 2014; Revised: 11 September 2015; Accepted: 12 September 2015

Abstract

The method introduced in this paper extends the trim-loss problem or also known as 2D rectangular SLOPP to the multiple sheet situation where N same size two-dimensional sheets have to be cut optimally producing demand items that partially or totally satisfy the requirements of a given order. The cutting methodology is constrained to be of the guillotine type and rotation of pieces is allowed. Sets of patterns are generated in a sequential way. For each set found, an integer program is solved to produce a feasible or sometimes optimal solution to the N-sheet problem if possible. If a feasible solution cannot be identified, the waste acceptance tolerance is relaxed somewhat until solutions are obtained. Sets of cutting patterns consisting of N cutting patterns, one for each of the N sheets, is then analysed for optimality using criteria developed here. This process continues until an optimal solution is identified. Finally, it is indicated how a given order of demand items can be totally satisfied in an optimal way by identifying the smallest N and associated cutting patterns to minimize wastage. Empirical results are reported on a set of 120 problem instances based on well known problems from the literature. The results reported for this data set of problems suggest the feasibility of this approach to optimize the cutting stock problem over more than one same size stock sheet. The main contribution of this research shows the details of an extension of the Wang methodology to obtain and prove exact solutions for the multiple same size stock sheet case.

Key words: Cutting stock, 2D rectangular SSSCSP, 2D rectangular SLOPP, exact solutions (for N sheets), Wang algorithm.

1 Introduction

Wäscher *et al.* [16] proposed a typology to categorise different problem formulations according to categorisation criteria. In this paper a two-dimensional cutting stock problem

*Corresponding author: Department of Computer Science and Information Systems, North-West University, Potchefstroom Campus, Private Bag X6010, Potchefstroom, 2530, South Africa, email: tjaart.steyn@nwu.ac.za

†**(Fellow of the Operations Research Society of South Africa)**, Department of Computer Science and Information Systems, North-West University, Potchefstroom Campus, Private Bag X6010, Potchefstroom, 2530, South Africa

http://dx.doi.org/10.5784/31-2-527

(2DCSP) is considered. This type of problem is normally encountered during the process of cutting a set of rectangular stock sheets like glass or wood into a set of smaller rectangular items in order to partially or fully satisfies a specified demand of the smaller items. It is accepted that guillotine type cuts are employed, that rotation of the small items is allowable and that the demand of small items cannot be satisfied by cutting only one stock sheet. An upper bound is specified on the quantity of each type of small item needed. It is also assumed that the stock sheets to be used are of the same size.

This problem is referred to by Wäscher *et al.* [16] as the two-dimensional single stock-size cutting stock problem (2DSSSCSP). They also defined the two-dimensional multiple stock-size cutting stock problem (2DMSSCSP) where different sizes of stock sheets and in variable quantities may in general be available. A well known elementary case is to cut one stock sheet to find a single cutting pattern producing the least waste. This type of problem is generally known as the trim-loss problem or the two-dimensional rectangular single large object placement problem (2D rectangular SLOPP) indicated as 2D-SLOPP in the rest of the paper.

The concept of the 2D-SLOPP is extended to a 2D-SLOPP over N same size sheets, called the N sheet two-dimensional single stock-size cutting stock problem (NS-2DSSSCSP) in this paper. Note that this is the same as 2DSSSCSP with N specified explicitly. This is a problem posed for utilizing the N sheets optimally by cutting items from the set of pieces in an order. For $N = 1$, this then reduces to the 2D-SLOPP. It is generally considered to be a NP hard problem to obtain the exact solution for $N > 1$.

To solve the 2DSSSCSP exactly either all the possible cutting patterns must be considered to establish optimality or some proof must be given that "sufficient" cutting patterns have been considered. An exact approach is proposed in this paper to maximize the utilization of a specified number of stock sheets without generally enumerating all possible cutting patterns. In §4 it is proved that it is possible to either guarantee optimality or give an indication that more cutting patterns must be generated to improve on the current solution. Although this approach is based on exactly N sheets, it is shown that the solutions can be utilized as part of a further search strategy to solve the 2DSSSCSP exactly to satisfy the full order of items. In the latter case it is assumed that sufficient stock sheets are available to fill the order.

The proposed solution process to establish the exact solution to the NS-2DSSSCSP consists of the systematic application of the following activities using an iterative process, namely

1. generate cutting patterns,
2. build and solve integer programming models, and
3. conduct optimality tests.

The NS-2DSSSCSP definition resembles the problem type of *output maximisation* defined by Wäscher *et al.* [16] whereby N large objects are supplied, which (in general) does not allow the accommodation of all small items. This means that there is generally a selection problem regarding the small items, and all the large objects have to be used. In this sense this problem (NS-2DSSSCSP) is an extension of the 2D-SLOPP.

The above mentioned approach of *output maximisation* is utilized by means of a search

strategy to establish the optimum number of sheets needed to fulfil the cutting of the specified demand set of all items in the order. This is explained in §5.2 of this paper. In this phase, it can be seen as an *input (value) minimisation* problem type (Wäscher *et al.* [16]) from the set of large objects. Thus, all small items are to be assigned to a selected number of large object(s) of minimal total "value" or area.

In §2 a brief literature review is given followed by §3 on notation and problem formulation. §4 contains a theoretical discussion and proofs to support the proposed algorithms. In §5 algorithms based on the theoretical work and some ideas from the literature are discussed. Empirical work and results follow in §6. §7 gives the conclusions followed by an overview of some further research possibilities based on the results.

2 Background and literature

In their paper on an improved typology of cutting and packing problems, Wäscher *et al.* [16] categorised a list of 413 papers dated up to December 2005 that they considered relevant to their description of cutting and packing problems. In the spreadsheet (dated February 28, 2012) linked to their paper, a list of 904 references is given. There are duplicates since a paper may be categorised into more than one category and when the duplicates are removed, there are 774 references. This illustrates that the field of cutting and packing continues to attract attention from researchers and quite a number of research papers are published regularly in this (broader) area.

A considerable number of research papers in the literature report on work regarding the trim-loss problem (SLOPP) whereby one stock sheet is cut into smaller demand items in a manner to minimize the waste. The spreadsheet from Wäscher lists 74 papers related to both two-dimensional and SLOPP. There are, however, a wide variety of approaches and definitions of this problem type. Some approaches allow rotation of items while others do not. Some accept pieces to be rectangular while others do not. Some follow a constrained (or staged) cutting approach while others do not. Some employ heuristics while others try to follow a more exact approach. When these characteristics are combined, only a few papers can be linked to the approach set out in this paper.

The process of solving a SLOPP in general needs some approach of generating and evaluating cutting patterns until optimality is reached or to stop the process with the solution at hand. Classical publications by Gilmore and Gomory [6, 7, 8] entail a series of papers on cutting stock problems where they initially concentrated on the one-dimensional problem. Their basic idea is to employ an unconstrained knapsack type of problem by utilizing dynamic programming strategies to establish favourable cutting patterns to be used in a (integer) linear programming (ILP) model. Other researchers like Christofides and Whitlock [4] employ pattern generation strategies based on tree search approaches. Most of these approaches involve mainly heuristics while the minority are exact of nature. The handling of a SSSCSP can generally be viewed as an iterative two-step process whereby the first step is to generate patterns (SLOPP approach) followed by a step of using these patterns as part of the input to solve the SSSCSP by means of a (integer) linear programming model.

Cintra *et al.* [5] report on various algorithms related to work done on two-dimensional cutting stock and strip packing problems. They employ dynamic programming and column generation techniques based on different instances of the cutting stock problem. These instances include a report on the Rectangular Knapsack (RK) problem which corresponds to the 2D-SLOPP according to the typology of Wscher *et al.* [16]. The algorithms they implemented are based on the recurrence formulas of the dynamic programming approach proposed by Beasley [3] which they combined with the discretization points of Herz [9]. The application of the Cintra algorithms to test problems gave good results timewise for the various instances of the 2D-SLOPP. In the next phase Cintra *et al.* [5] address the 2D-SSSCSP by means of a column generation technique based on the ideas initially proposed by Gilmore and Gomory.

As an alternative approach to dynamic programming, Wang [15] proposed two algorithms to establish a cutting pattern with least waste over a single sheet, thus a SLOPP. In her algorithms Wang employs a parameter $\beta \in [0, 1]$ as a proportion of the stock sheet area and uses it to discard patterns with waste proportion exceeding β during the process of cutting pattern generation. She generates all feasible cutting patterns by successively combining rectangles in a bottom-up implicit enumeration algorithm whilst conforming to constraints like allowable size, quantity and waste percentage specification. By increasing β from some starting value in successive iterations, the number of different patterns generated is kept as low as necessary.

Based on the work of Wang, a number of researchers like Oliveira and Ferreira [11] and Vasko [13] reported refinements to the algorithms in order to reduce computational time. Vasko and Bartkowski [14] used a modified version of Wang's first algorithm to experiment with the initial value and step size of β to obtain optimal solutions to 265 difficult cutting stock problems. They reported a fair amount of success with a step size of 0.002 and an initial value of zero for the test problems. About all SLOPP approaches identifiy only the (one) pattern (or alternatives) associated with the minimal waste.

Amaral and Wright [1] proposed an algorithm also based on the work of Wang that enhances the process of cutting pattern generation. By noting equivalences in combining rectangles vertically and horizontally and noting the legitimate rotation of rectangles, it is possible to eliminate certain combinations early on. They combined this elimination process with ordered data structures to further reduce the combinatoric effect. An added advantage of their approach, especially for the purposes of the research reported on in this paper, is that their algorithm generates (and stores) a list of all possible cutting patterns satisfying the β specification in a very efficient manner. In §6 some empirical results will be given based on the algorithm proposed by them.

During algorithm testing, Beasley [2] generated a set of 12 problems of the 2DMSSCSP type. Each problem considers cutting a set of patterns from multiple stock size rectangles and a list of small items for which the orientations are considered to be fixed. Beasley did some empirical work and reports that the algorithms produce a relatively low amount of waste but does not establish least waste comparisons. As an illustration of the waste levels found, a waste area percentage of 5.89% for problem 12 is reported. This was achieved by using 3 stock rectangle types. This set of problems has been used during the testing of the algorithms proposed in this paper.

3 Notation and problem formulation

A 2DCSP model is first considered to address the more formal part of this research, where an adequate number of stock sheets are available and the set of small demand items to be cut is specified. If all possible cutting patterns are available whereby demand items can be cut from a single stock sheet, the following integer linear program can be defined to compute an optimal solution for this 2DCSP. The objective is to minimize the total waste. Mathematically it can be formulated as

$$\text{minimize} \quad \sum_{j=1}^{n} w_j x_j \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} I_{ij} x_j = I_i, \qquad i = 1, \ldots, m, \tag{2}$$

$$x_j \text{ non-negative integers}, \qquad j = 1, \ldots, n, \tag{3}$$

where

x_j is the number of times pattern j is to be cut,
w_j is the (total) waste implied by pattern j,
I_{ij} is the number of times demand item i is cut by pattern j,
I_i is the quantity of demand item i specified,
n is the number of all possible cutting patterns, and
m is the number of demand item types in the order.

This model can be applied to the (2D)SSSCSP and adapted for the (2D)MSSCSP. In general it is time consuming and in some cases impractical to generate all possible cutting patterns even for a single stock sheet type. This is an important reason why researchers often employ heuristics or consider relaxations for the problem. If multiple stock sheet types are involved, it complicates matters further. In this research a single sheet type is considered and the following notation is adopted to address the NS-2DSSSCSP. Let

N specify the number of same size stock sheets to be used.
S_A specify the area of the sheet, thus $S_A = H \times L$, where H and L are the dimensions of the sheet,
β specify the *waste* proportion per sheet allowed for in the process of generating patterns according to the approach of Amaral and Wright [1] based on the initial ideas introduced by Wang [15][1],
$W_\beta = \{w_j^\beta \mid w_j^\beta \leq \beta S_A$ and $w_j^\beta$is the total waste for pattern $j\}$, and
$\beta_1 = \min_{0 \leq \beta \leq 1} \{\beta \mid W_\beta \neq \phi\}$.

Consider a special case of the NS-2DSSSCSP where the patterns used are generated based on a $\beta \geq \beta_1$. Define a model $SW(N, \beta)$ with the objective to minimize the total waste

[1] All cutting patterns with a (total) *waste* $\leq \beta S_A$ are thus kept while the others are discarded.

over N sheets and pattern waste based on β. Mathematically it is expressed as

$$\text{minimize} \qquad Z(N,\beta) = \sum_{j=1}^{n^\beta} w_j^\beta x_j \tag{4}$$

$$\text{subject to} \qquad \sum_{j=1}^{n^\beta} I_{ij} x_j \leq I_i \qquad i = 1,\ldots,m \tag{5}$$

$$\sum_{j=1}^{n^\beta} x_j = N \tag{6}$$

$$x_j \text{ non-negative integers}, \qquad j = 1,\ldots,n^\beta \tag{7}$$

where

x_j is the number of times pattern j is to be cut,
$w_j^\beta \in W_\beta$ is the (total) waste implied by pattern j,
I_{ij} is the number of times demand item i is cut by pattern j,
I_i is the quantity of item i specified,
m is the number of demand item types,
N is the specified number of (same size stock) sheets to be cut, and
n^β is the number of all cutting patterns with (total) waste within β specifications.

The first $m+1$ constraints in the model constrain the number of demand items to be cut not to exceed the demand specified and also the number of stock sheets to be used. Thus, the above model adheres to the problem type of *output maximization* as defined by Wäscher *et al.* [16]. For values of β and N that render $SW(N,\beta)$ feasible, such $SW(N,\beta)$'s have optimal solutions with objective function values of (say) $Z^*(N,\beta)$. For non-decreasing sequences of values for β, $Z^*(N,\beta)$ forms a non-increasing sequence of values. In practice strictly increasing values of β will be considered. If β is sufficiently large the sequence converges to the optimal objective function value $Z^*(N)$ that exploits the N sheets optimally for the given set of demand items. There thus exists a (minimum) $\beta^* \geq \beta_1$ that is large enough such that $Z^*(N,\beta) = Z^*(N)$ for $\beta \geq \beta^*$. It should be noted that $Z^*(N,\beta)$ gives the optimal objective function value to $SW(N,\beta)$, but often it may not be the optimal objective function value to the NS-2DSSSCSP. It will, however, be clear from the context. The problem to identify $Z^*(N)$ is a generalization of the (single sheet) trim-loss (2D-SLOPP) problem to the N sheet trim-loss problem. (Note that for all $\beta \geq \beta_1$, $Z^*(1) = \min\{w_j^\beta \mid w_j^\beta \in W_\beta\}$).

4 Theoretical discussion and proofs

One method to establish the optimum solution of the cutting stock problem described here, is to have all possible cutting patterns available and to solve the relevant integer linear program. An alternative is to utilize the current information (or patterns) to prove optimality or to indicate that more patterns are needed. These additional patterns are generated by employing a new larger β. Thus, the theorems and corollaries given in this section enable the decision maker to either establish that $Z^*(N) = Z^*(N,\beta)$ and

optimality is reached or give an indication that the set of cutting patterns have to be extended by increasing the current value of β. For the ease of the exposition of the theory below it is assumed that the solutions mentioned exist and β has appropriate values.

Theorem 1 constitutes on the one hand optimality conditions for the specified NS-2DSSSCSP based on the current information in the form of patterns with their associated waste that were generated by utilising a certain β. If these optimality conditions do not hold, it means that the solution associated with $Z^*(N,\beta)$ based on the current β-value is not optimal to the NS-2DSSSCSP. Thus the β-value must be increased in order to generate additional patterns and retesting the conditions for optimality so that $Z^*(N,\beta) = Z^*(N)$ could ultimately be confirmed. On the other hand, if optimality to the problem is confirmed in some way, it can be claimed that these conditions must hold. Thus, this theorem forms the basis of the theoretical part of this research and plays a central role in the implementation of the algorithm developed. It can also be seen that although the theorem is stated in terms of N to correspond to the so-called NS-2DSSSCSP, the value N is a generalization and can thus be applied to r, where $r \leq N$ in the NS-2DSSSCSP.

Theorem 1
For a given β, $Z^*(N,\beta) = Z^*(N)$ if and only if, for all $r \in \{1,2,\ldots,N-1\}$, with $N \geq 2$, $Z^*(r) + (N-r)\beta S_A \geq Z^*(N,\beta)$.

Proof: Consider the if-part first and let $Z^*(N,\beta) > Z^*(N)$ implying that $Z^*(N,\beta)$ is not optimal for the NS-2DSSSCSP. Then a $\beta' > \beta$ must exist such that $Z^*(N,\beta) > Z^*(N,\beta')$. This implies that the optimal solution for $SW(N,\beta')$ must have at least one pattern associated with a "new" pattern generated by the relaxation to β' (*i.e.* $\beta S_A <$ `waste of new pattern` $\leq \beta'.S_A$). Let there be $(N-r)$ such new patterns. The other r patterns in the solution have associated `waste` $\leq \beta S_A$. It can be argued that, for the solution associated with β',

$$
\begin{aligned}
Z^*(N,\beta) &> Z^*(N,\beta') \\
&= \Big(\sum \texttt{waste of } r \texttt{ patterns with waste} \leq \beta S_A\Big) \\
&\quad + \Big(\sum \texttt{waste of } (N-r) \texttt{ new patterns with } \beta S_A < \texttt{waste} \leq \beta' S_A\Big) \\
&\geq Z^*(r) + \Big(\sum \texttt{waste of } (N-r) \texttt{ new patterns with } \beta S_A < \texttt{waste } \leq \beta' S_A\Big) \\
&> Z^*(r) + (N-r)\beta S_A
\end{aligned}
$$

This proves that an r exists such that the condition does not hold.

Consider the only-if-part next and say that an $r \in \{1,2,\ldots,N-1\}$ with $N \geq 2$ exists such that $Z^*(r)+(N-r)\beta S_A < Z^*(N,\beta)$. Adjust the model $SW(N,\beta)$ by removing the number of demand items in the model that are included in the patterns used in the solution that gives $Z^*(r)$ and thus adjusting the upperbounds specified on those demand items. Solve the adjusted model $SW^a(N-r,\beta)$ and find the objective function value $Z^a(N-r,\beta)$. The combination of patterns that form the solutions giving the objective function values $Z^*(r)$ and $Z^a(N-r,\beta)$, is a feasible solution to the original model $SW(N,\beta)$ and thus

one can argue that $Z^*(N,\beta) > Z^*(r) + (N-r)\beta S_A \geq Z^*(r) + Z^a(N-r,\beta)$. This proves that $Z^*(N,\beta)$ is not optimal. This completes the proof. □

Based on the concepts in Theorem 1, the attributes of the left hand sides of the optimality criterion is considered. For a given β, these left hand sides form a non-decreasing sequence in r for the r values under consideration. The value of the following theorem lies in the fact that it is sufficient to only check whether $Z^*(k-1) + \beta S_A \geq Z^*(k,\beta)$ for a certain k to confirm or reject optimality of $Z^*(k)$. This follows from the characteristics of the sequence which is proved below.

Theorem 2
The left hand sides of the optimality criterion used in Theorem 1 for a certain value of β viz. $Z^*(r) + (N-r)\beta S_A\{<,\geq\}Z^*(N,\beta)$ for $r \in \{1,2,\ldots,N-1\}$, form a sequence that is non-decreasing in r.

Proof: Consider a value of k such that $k \in \{3,4,\ldots,N+1\}$. If $Z^*(k-1)$ is optimal, one can argue that $Z^*((k-1)-1)+\beta S_A \geq Z^*(k-1,\beta) = Z^*(k-1)$. That is, $Z^*(k-2)+2\beta S_A \geq Z^*(k-1) + \beta S_A$, proving that the sequence is non-decreasing. □

Corollary 1 gives some further information that follows from Theorem 1 and that may be useful in the development of algorithms.

Corollary 1
When optimal solutions are produced by applying the procedure suggested in the proof of Theorem 1, a sequence of optimal solutions $Z^*(r,\beta) = Z^*(r)$ for $r \in \{1,2,\ldots,N-1\}$ are produced for the rS-2DSSSCSP.

Different methods are possible to establish underestimates $Z^U(r)$ for $Z^*(r)$ with $Z^U(r) \leq Z^*(r) \leq Z^*(r,\beta)$ and $\beta_1 \leq \beta \leq 1$. Note that since $Z^*(r-k) + Z^*(k) \leq Z^*(r)$, then $Z^U(r) = Z^*(r-k)+Z^*(k)$ is an underestimate for $Z^*(r)$ for situations where $Z^*(r-k)$ and $Z^*(k)$ (or good underestimates of them) are known. Utilizing this may save considerable computational time. Formally, the following corollary follows directly from Theorem 1.

Corollary 2
If $Z^U(r) + (N'-r)\beta S_A \geq Z^*(N',\beta)$ for all r-values with $r \in \{1,\ldots,N'-1\}$ and $N' \in \{2,\ldots,N\}$, one can conclude that $Z^*(N',\beta) = Z^*(N')$.

5 Agorithms

The knowledge from Theorem 1 and 2 together with Corollary 1 and 2 can be used to develop the following procedure.

5.1 An algorithm to establish $Z^*(N)$

Algorithm 1 (NS-algorithm) has been devised to establish $Z^*(N)$ for a specified $N > 1$ for the NS-2DSSSCSP without necessarily generating all possible cutting patterns. $N = 1$ constitutes the trim-loss case (2D-SLOPP). It is assumed that β is large enough so that $Z^*(s, \beta)$ exists for all $s \leq N$. It is also assumed that $Z^*(N, \beta) > \beta S_A$ since the alternative would imply optimality for that N.

Algorithm 1: NS-algorithm: Generalized N sheet trim-loss algorithm

```
STOP = False
while !STOP do
    Generate Patterns W_β (Read W_β from file according to β)
    Compute Z*(N, β) for model SW(N, β)
    if Stop_reason<>"Full_Fit" then
        WasteSum[N] = Z*(N, β)
        s = N − 1; β_flag= True
        while s > 0 and β_flag do
            Compute Z*(s, β) for model SW(s, β)
            WasteSum[s] = Z*(s, β)
            k = 0
            while k < (N − s) and β_flag do
                β_flag = (WasteSum[s] + (N − s − k) ∗ βS_A ≥ WasteSum[N − k])
                k = k + 1
            if β_flag then
                s = s − 1
                if s ≤ 0 then
                    STOP = True; Z*(N) = Z*(N, β); STOP_reason="opt. proved"
            else
                Increase β
    else
        STOP = True; STOP_reason = "Full_Fit"
Report results.
```

For the empirical work presented in §6, the algorithm of Amaral and Wright [1] is used to generate cutting patterns. Although not explicitly reflected in the NS-algorithm, underestimates are computed and utilized in the developed program.

5.2 Establishing an optimal number of sheets

As an application and an extension to the NS-algorithm, Algorithm 2 as an order cutting algorithm (OC-algorithm) was devised to establish the optimum (minimum) number of sheets (or the number giving the minimum total waste area) needed to satisfy the given 2DSSSCSP. The OC-algorithm identifies the optimal number of sheets (N) by employing the NS-algorithm in a structured manner for different N-values.

Define T_A as the total area of the demand items. When the NS-algorithm is applied to a N-sheet problem (with $N \geq 1$) and the NS-2DSSSCSP is solved, a value $T_A^*(N)$ is obtained denoting the total area of the demand items accommodated on the N sheets plus a residual waste of $Z^*(N)$. It thus holds that $NS_A = T_A^*(N) + Z^*(N)$. If $T_A^*(N) < T_A$ it

holds that $NS_A < T_A + Z^*(N)$ implying that the N-value is not optimal (too low) and gives a lower bound on the optimal N. If $T_A^*(N) = T_A$, it holds that $NS_A = T_A + Z^*(N)$ implying that the N-value is either optimal or too large. It gives an upper bound on the optimal N and all the demand items are accommodated on the N sheets, giving a so-called full-fit.

Some of the variables being used in the OC-algorithm to solve the 2DSSSCSP are explained next:

$\texttt{MAX}_N$ is an upper bound on the optimal number of sheets needed giving a full-fit for the 2DSSSCSP. The initial $\texttt{MAX}_N$ may be established by means of a heuristic approach. The OC-algorithm strives to find the minimum $\texttt{MAX}_N$.

$\texttt{MIN}_N = LB_N - 1$ where LB_N is a lower bound on the optimal number of sheets needed giving a full-fit for the 2DSSSCSP. The initial LB_N may be computed based on a theoretical approach, *e.g.* $\lceil T_A/S_A \rceil$. The OC-algorithm strives to find the maximum $\texttt{MIN}_N$ and thus minimizes the area of the demand items not accommodated on the 'first' $\texttt{MIN}_N$ sheets. $\texttt{MIN}_N < \texttt{MAX}_N$ by definition.

The STOP-criterion is activated when $\texttt{MAX}_N = \texttt{MIN}_N + 1$. In this case $\texttt{MAX}_N$ will be the least number of sheets needed to satisfy the 2DSSSCSP, giving a full-fit. If $\texttt{MIN}_N$ is such that for $Z^*(\texttt{MIN}_N)$ the demand items not included in this solution do fit onto one additional sheet, it gives the solution with the best utilization possible in terms of waste regarding the first $\texttt{MIN}_N$ sheets and thus maximizes the unused area on the last sheet.

The implementation of the OC-algorithm employs a binary search approach to establish the optimal values for $\texttt{MAX}_N$ and $\texttt{MIN}_N$.

Algorithm 2: OC-algorithm: An order cutting algorithm

```
Input: The 2DSSSCSP and initial values for MAX_N and MIN_N,
N = ⌈(MAX_N + MIN_N)/2⌉
while MAX_N > MIN_N + 1 do
    Execute the NS-algorithm to establish Z*(N)
    if STOP_reason = "Full_Fit" then
        MAX_N = N
    else
        MIN_N = N
    N = ⌈(MAX_N + MIN_N)/2⌉
Output: The optimal MAX_N and MIN_N.
```

6 Empirical work

In this section results are given that empirically evaluate the performance of the algorithms reported on in this paper. The data set used is a set of 12 problems (indicated by $B1$ to $B12$) as defined by Beasley [2]. For each of these 12 problems a set of 10 stock sheet sizes is available. Beasley considered the 2DMSSCSP type of problem and did empirical work on these 12 problems. The research in this paper focuses on the 2DSSSCSP type of problem and for experimental purposes each of these 12 problems is specified as a range of 10

2DSSSCSP's by considering the set of demand items as specified for each of the stock sheet sizes separately. The first 2DMSSCSP, B1, is thus associated with 10 2DSSSCSP's named $B1\_1, \ldots, B1\_10$ and similarly for $B2, B3, \ldots, B12$. Finally, the resulting 120 problems are indicated by $B1\_1; \ldots; B1\_10; B2\_1; \ldots; B12\_1; \ldots; B12\_10$. The code developed to test the algorithms and approaches, were written in the programming language C# (Microsoft Visual Studio 2008) and implemented on a HP Notebook computer with 2 Gb RAM and a 2.40 GHz Intel® core™2 Duo CPU.

6.1 System tested on the trim-loss problem

The first phase of the experimental work was to implement the algorithm of Amaral and Wright [1]. A program was developed and tested [12] and compared to the results reported by Amaral and Wright for the same data set. Table 1 compares the results for problem $B12$ as a trim-loss problem from the test bed for the CSA and EWA algorithms as reported by Amaral and Wright [1] and the TSA-algorithm as implemented for this research.

Problem	Time (sec)			Total waste		
	CSA	EWA	TSA	CSA	EWA	TSA
$B12\_1$	0.032	0.068	0	1234	1234	1234
$B12\_2$	0.031	0.207	0.03	1586	1586	1586
$B12\_3$	0.048	0.428	0	2310	2310	1476
$B12\_4$	0.047	0.287	0.02	1256	1256	1256
$B12\_5$	0.048	0.338	0.01	1053	1053	1053
$B12\_6$	0.071	1.149	0.02	1203	1203	1203
$B12\_7$	0.104	2.514	0	571	571	571
$B12\_8$	0.022	0.054	0	1041	1041	1041
$B12\_9$	0.549	32.659	0.11	410	410	410
$B12\_10$	0.1	3.264	0.03	203	203	203

Table 1: *A comparison of the results by Amaral & Wright and this research.*

Remarks:

- The time reported in Table 1 indicates only the time needed by the algorithms for producing the optimal pattern. The performance of the TSA-algorithm in terms of time seems to be good when compared to the time measurements as indicated by Amaral and Wright.
- The waste generated by TSA for problem $B12\_3$ for the optimal (trim-loss) pattern differs from the waste as reported by Amaral and Wright. The waste of 1476 reported by TSA corresponds with a legitimate pattern and it (TSA) also generated a pattern with the same waste of 2310 (as its second best pattern). It is unclear why Amaral and Wright did not find the optimal solution.

6.2 Feasibility of generating all cutting patterns

The next phase was to test the use of the TSA-algorithm on the whole set of 120 problems to generate cutting patterns. For each problem the algorithm starts with $\beta = 0.0$ and

increases β with intervals of 0.01. The patterns generated are saved in a text file to be used in later experimentation. Although generally a time limit of two hours is set to generate the cutting patterns, extra time was allowed in some cases. Table 2 gives results regarding each of the problems. The first row in each triplet (Beta) contains the maximum value for β reached to generate cutting patterns within the allowable time; the second row (Time) indicates the total time (in seconds) needed to generate patterns from $\beta = 0.0$ up to the reported β (when stopped) and the third row (#Patt) indicates the number of patterns generated for the last β reported for the given problem. The TSA-algorithm generated all patterns (up to $\beta = 1.0$) for 87 of the 120 problems in the time allowed. From these 87 problems, 65 took less than 10 minutes. Only 5 of the remaining 22 took more than an hour. There are thus 33 problems for which not all (up to $\beta = 1.0$) cutting patterns could be generated in the time allowed.

6.3 Empirical work with the NS-algorithm

The main objective of the experiments here is to establish the feasibility of finding $Z^*(N)$ by solving the model $SW(N, \beta)$ for a given NS-2DSSSCSP and thus implementing the NS-algorithm. A procedure that calls CPLEX [10] in order to solve the model $SW(N, \beta)$ was developed in the programming language C# to determine $Z^*(N)$ given the set of demand items, the stock sheet dimensions and N. In solving a mixed integer program, CPLEX uses a tolerance option, *epgap*, with a default value of 0.0 which specifies the stop criterion for the relative tolerance on the gap between the objective of the best current feasible solution and a lower bound for the optimal solution. This gap is traditionally called the integrality gap. In cases where CPLEX cannot solve the model easily, it is necessary to manage this gap specification. Therefore, the time limit option, *tilim*, that determines the amount of time in seconds that CPLEX will devote to solve a problem, is set to 5 (*i.e.* 5 seconds). Whenever CPLEX finds a feasible solution to $SW(N, \beta)$, but cannot establish optimality within the given time limit, *epgap* is increased by 0.001, and the process is repeated with the new gap tolerance.

Table 3 illustrates and summarizes the results for the model $SW(N = 387, \beta)$ and the problem $B12\_3$. The third column (Time for β) indicates the time taken by the TSA-algorithm to generate the patterns for the β indicated.

The total time taken by the NS-algorithm to compute $Z^*(N = 387) = 2360147$ is 58.17 seconds and a β-value of 0.49 is sufficient to prove optimality. Similar experimentation was performed for different values of N on all the problems in the problem set. See §6.4 for further demonstration.

6.4 Empirical work with the OC-algorithm

As an extension to the NS-algorithm, the OC-algorithm was devised and developed as described in §5.2 to establish the optimal number of sheets to fulfill an order of demand items using same size sheets. This was applied to problem $B12\_3$ and the results are shown in Table 4. The initial value (319) for $\texttt{MIN}_N$ is computed theoretically by using the total area of the demand items and the sheet size. The initial value (455) for $\texttt{MAX}_N$ is computed by means of an available glass cutting program based on heuristic approaches.

		$B1$	$B2$	$B3$	$B4$	$B5$	$B6$	$B7$	$B8$	$B9$	$B10$	$B11$	$B12$
	Beta	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**0.75**	1.00	1.00
Nr1	Time	40.69	0.4	2403.49	142	0.38	2.67	707.48	5206.55	791.6	**6133.18**	37.43	1.48
	#Patt	518	123	5830	935	123	411	1254	5333	3332	**3667**	1340	351
	Beta	**0.35**	**0.20**	1.00	1.00	1.00	**0.05**	1.00	**0.08**	1.00	1.00	**0.34**	1.00
Nr2	Time	**7003.88**	**7080**	3164.23	9.43	13.81	**6720**	35.69	**10080**	5264.08	96	**7058.96**	166.43
	#Patt	**3903**	**14198**	7664	485	669	**16749**	539	**9619**	8553	1029	**11640**	3868
	Beta	1.00	**0.10**	**0.10**	1.00	1.00	1.00	1.00	**0.35**	1.00	1.00	**0.60**	1.00
Nr3	Time	261.32	**12660**	**5040**	215	1065.53	38.57	785.93	**12859.68**	630.56	149	**7483.49**	3.82
	#Patt	751	**16724**	**18118**	1251	3567	1178	1204	**8690**	3055	1205	**9142**	582
	Beta	1.00	**0.68**	1.00	**0.55**	1.00	1.00	1.00	1.00	1.00	1.00	**0.39**	1.00
Nr4	Time	0.19	**6865.77**	6596.39	**6220.49**	0.29	766.51	424	0.57	39.46	0.64	**6912.15**	20.77
	#Patt	28	**8804**	10787	**4145**	80	3461	1114	103	1493	115	**11300**	897
	Beta	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**0.70**	1.00	1.00	1.00	1.00
Nr5	Time	66.71	58.4	94	15.53	696.2	64	2786.03	**7147.34**	4199.9	0.56	30.49	9.55
	#Patt	507	1218	2048	388	3003	1392	2211	**4872**	4545	112	1020	826
	Beta	1.00	**0.46**	1.00	1.00	**0.10**	1.00	1.00	**0.20**	**0.08**	1.00	1.00	1.00
Nr6	Time	654.9	**7216.03**	1735.97	0.26	**5820**	340.33	840	**7020**	**6960**	15.14	1165.4	35.78
	#Patt	1269	**11455**	5452	77	**16998**	3295	1295	**10876**	**12526**	376	3022	1484
	Beta	1.00	**0.15**	**0.20**	1.00	**0.17**	**0.08**	1.00	1.00	**0.06**	1.00	1.00	1.00
Nr7	Time	1.98	**6600**	**6420**	0.14	**7620**	**5100**	1355.15	108	**5880**	19.67	2485.9	358.52
	#Patt	138	**18193**	**18636**	39	**17793**	**15637**	1738	1098	**8102**	430	4159	3497
	Beta	1.00	1.00	**0.14**	1.00	**0.18**	**0.23**	1.00	1.00	**0.85**	1.00	1.00	1.00
Nr8	Time	4.78	0.5	**7020.44**	583.48	**6120**	**6813.99**	0.43	439	**7378.95**	0.17	1602.02	1.13
	#Patt	207	131	**18315**	1819	**16437**	**17867**	76	1709	**7981**	25	3120	309
	Beta	1.00	1.00	1.00	1.00	**0.14**	1.00	1.00	1.00	1.00	1.00	1.00	**0.09**
Nr9	Time	0.2	160	86	0.19	**10320**	0.71	72	0.63	1.74	1.02	52.07	**6960**
	#Patt	29	1772	1966	60	**21466**	179	551	122	257	166	1150	**26919**
	Beta	1.00	1.00	1.00	1.00	**0.90**	1.00	1.00	1.00	**0.55**	1.00	**0.66**	**0.16**
Nr10	Time	0.25	321	226	0.35	**7128.12**	7827.87	0.23	1.48	**6367.32**	0.2	**7058.35**	**9240**
	#Patt	40	2464	2283	89	**8091**	7543	60	202	**11349**	54	**7940**	**9311**

Table 2: *A summary of the TSA results for the data set from Beasley [2]*

Beasley12_3			$N = 387$	Total time needed: 58.17 seconds
β	# patterns	Time for β	Waste	Comments
0.32	328	00:00.03	Infeasible	Too few patterns; no feasible solution
0.33	348	00:00.03	2842119	Feasible solution; Need to increase β
0.34	361	00:00.03	2842119	
0.35	375	00:00.03	2842119	
0.36	391	00:00.03	2842119	
0.37	412	00:00.03	2842119	
0.38	424	00:00.03	2842119	
0.39	437	00:00.04	2842119	
0.4	454	00:00.04	2842119	
0.41	470	00:00.04	2842119	
0.42	480	00:00.04	2842119	
0.43	494	00:00.04	2842119	
0.44	501	00:00.04	2393159	Improved waste; Still need to increase β
0.45	513	00:00.04	2393159	
0.46	522	00:00.04	2393159	
0.47	528	00:00.04	2393159	
0.48	533	00:00.04	2393159	
0.49	541	00:00.04	2360147	Improved waste; $Z^*(N)$ confirmed; STOP

Table 3: *The NS-algorithm applied to problem B12_3 (with $N = 387$).*

Table 4 summarizes the results comparable to those in Table 3 for each N value in the order required by the OC-algorithm, thus sequentially from $N = 387$ to $N = 439$ when the stop criterion is reached. In the first row (Result) an entry of "Optimal" means that the NS-algorithm was able to prove utilization optimality for the N requested and an entry of "F.Fit" means that a full-fit was established for the specific N value. In the second and third rows ($\texttt{MIN}_N$ and $\texttt{MAX}_N$) the numbers indicate the values related to the resulting lower and upper bounds respectively after the application of the NS-algorithm. A minimum of 440 sheets are needed to fulfill the order of demand items. By inspecting the cutting patterns associated with $Z^*(N = 439)$, computations show that the remaining demand items correspond to a feasible cutting pattern for one sheet. It can therefore be concluded that by optimizing over $N = 439$ sheets, the area not utilized on the 440^{th} sheet is maximized. The fourth row in the table (Time) demonstrates the performance of the process in terms of time needed to solve the problem. The maximum β-value required to solve the problem is 0.51.

In order to further explore the wider applicability of the NS-algorithm and its extension, the OC-algorithm was applied to all 120 problems in the problem set. Table 2 indicates that for 33 of these problems the TSA-algorithm could not generate all the patterns up to $\beta = 1.0$ within the time limit. These 33 problems were investigated further and can be divided into two sets, *i.e.* a set of 12 problems with sufficient β-values (and patterns) to prove optimality by means of the OC-algorithm and a set of 21 problems requiring larger β-values (and patterns). Table 5 gives the results of the 12 problems. The sixth column (TSA Max β) gives the largest β-value for which patterns were generated in the allotted time. The seventh column (OC-alg Max β) gives the final β-value employed by the OC-algorithm to prove optimality while the eighth column (Time for β sufficient) gives the time needed by the TSA-algorithm to generate patterns up to the β-value sufficient

β	$N = 387$	$N = 421$	$N = 438$	$N = 447$	$N = 443$	$N = 441$	$N = 440$	$N = 439$
Result	Optimal	Optimal	Optimal	F.Fit	F.Fit	F.Fit	F.Fit	Optimal
$\texttt{MIN}_N$	387	421	438	438	438	438	438	439
$\texttt{MAX}_N$	455	455	455	447	443	441	440	440
Time	00:58.17	01:05.86	01:10.05	00:10.32	00:10.07	00:10.11	00:09.99	01:11.61
0.32	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
0.33	2842119	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
0.34	2842119	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
0.35	2842119	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
0.36	2842119	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
0.37	2842119	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
0.38	2842119	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
0.39	2842119	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
0.4	2842119	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
0.41	2842119	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
0.42	2842119	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
0.43	2842119	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
0.44	2393159	3235135	3656123	3878999	3779943	3730415	3705651	3680887
0.45	2393159	3235135	3656123	3878999	3779943	3730415	3705651	3680887
0.46	2393159	3235135	3656123	3878999	3779943	3730415	3705651	3680887
0.47	2393159	3235135	3656123	3878999	3779943	3730415	3705651	3680887
0.48	2393159	3235135	3656123	3878999	3779943	3730415	3705651	3680887
0.49	2360147	2956417	3377405	3600281	3501225	3451697	3426933	3402169
0.5		2772481	2979269	3184481	3085425	3035897	3011133	2991787
0.51								2991787

Table 4: *A summary of the results for the OC-algorithm applied to problem B*12_3.

to prove optimality. The second last column ($\#N$) indicates the number of times the OC-algorithm called the NS-algorithm with a specified N. The last column (Time for OC-algorithm) indicates the time needed for the OC-algorithm. Only problem B11_10 took more than 2 hours and the others took less than 1 hour each. It shows that even for problems for which it is time consuming to generate cutting patterns, the "readily" available patterns are often sufficient to prove optimality and in this case brings the total of 'solved' problems to 99 out of the 120.

The set of 21 problems not solved to optimality by means of the OC-algorithm, was analyzed further. The main objective was to get an indication as to "how far" one can get with the readily available patterns. In order to do this, the NS-algorithm was employed in a structured way with different values for N for each problem to establish the results as reported in Table 6. The first and second columns (Initial *LB* and Initial *UB*) correspond to the (initial) *LB* and (initial) $\texttt{MAX}_N$ respectively as discussed earlier. As another lower bound the fifth column (Max N Opt) gives the maximum N for which optimality of utilization is proved using the available patterns. The sixth column (Max β N Opt) indicates the maximum β with associated patterns sufficient to prove optimality for the corresponding *Max* β *N Opt* value. The fourth column (Min N F.Fit) indicates the minimum N for which a full-fit is established by the NS-algorithm. For only two problems, B6_2 and B6_8, it was not possible to establish a full-fit utilizing the available patterns. The seventh column (% not utilized) gives the percentage of the area of demand items not included as part of the *Max N Opt* sheets while the eighth column (N not placed) gives a theoretical lower bound on the number of sheets still needed to cut these demand items. The last column (%Gap) is calculated by ((Column 4 - Column 9)/ Column 4)×100. Column 9 (Final

Problem	Initial $\texttt{MIN}_N$	Initial $\texttt{MAX}_N$	Min $\texttt{MAX}_N$	Max $\texttt{MIN}_N$	TSA Max β	OC-alg Max β	Time for β sufficient	$\#N$	Time for OC-algorithm
$B1\_2$	49	56	54	53	0.35	0.30	4091.63	3	1730.3
$B2\_4$	133	152	147	146	0.68	0.36	1892.49	4	2168.9
$B3\_7$	188	201	194	193	0.20	0.12	540.36	4	631.7
$B4\_4$	51	55	54	53	0.55	0.16	87.98	2	479.1
$B5\_10$	132	157	146	145	0.90	0.33	933.86	5	847.5
$B8\_5$	142	186	176	175	0.70	0.41	2193.16	6	837
$B9\_8$	230	310	301	300	0.85	0.45	2289.94	7	3223.5
$B9\_10$	226	273	257	256	0.55	0.47	4713.04	6	1858.7
$B10\_1$	52	68	61	60	0.75	0.38	1375.31	4	536.8
$B11\_3$	123	133	128	127	0.60	0.26	1126.9	3	300.2
$B11\_4$	119	131	130	129	0.39	0.29	3035.5	4	1333.9
$B11\_10$	126	148	145	144	0.66	0.50	4120.73	5	7232

Table 5: *The OC-algorithm applied to the set of 12 problems (with $\beta < 1.0$) not solved to optimality.*

LB), calculated by (Column 5 + Column 8), serves as a measure of success by utilizing the available patterns. The gap seems to be useful since 16 of the problems have a gap of 3.85% or less with 6 of these with a gap of even less than 1%. These results suggest that the NS-algorithm may be valuable in heuristic search techniques like greedy search approaches.

7 Conclusions

The TSA-algorithm based on the work of Amaral and Wright [1] is given to enable the empirical research with efficient code. As a test bed, the problems as reported by Beasley [2] were adapted to give a total of 120 2DSSSCSP's. This implementation of the TSA-algorithm enabled the generation of all possible cutting patterns for 87 of the 120 test problems within the time allowed and is thus regarded as useful.

Table 3 illustrates the application of the NS-algorithm to solve the problem $B12\_3$ exactly for a specified number of $N = 387$ sheets. It took 58.17 seconds with a maximum β value of 0.49 sufficient to prove optimality. Thus, in this case, the number of cutting patterns generated could be limited. The NS-algorithm was extensively tested on all 120 problems with a range of values specified for N according to the requirements of the OC-algorithm. The OC-algorithm utilizes the NS-algorithm to solve the problem of finding the optimal number of sheets for cutting the total order. Table 4 illustrates the OC-algorithm applied to problem $B12\_3$. The optimal number of sheets needed for this problem is $N = 440$ and it is also established that the area not used on the last, *i.e.* the 440^{th} sheet, is maximized. The time needed for each specified N (the NS-algorithm) does not exceed 72 seconds showing that the process is feasible for this problem. The lowest β value sufficient to solve the problem to optimality is 0.51.

For 82.5% of the 120 test problems the application of the OC-algorithm results in the optimal number of sheets (and cutting patterns) and in that sense demonstrates the generalization of the trim-loss problem for a single sheet to N sheets (without necessarily

Problem	Initial *LB*	Initial *UB*	Min *N* F.Fit	Max *N* Opt	Max β *N* Opt	% not utilized	*N* not placed	Final *LB*	% Gap
*B*2_2	108	125	117	101	0.2	10.43	12	113	3.42
*B*2_3	96	104	99	95	0.09	2.99	3	98	1.01
*B*2_6	120	138	134	128	0.41	2.26	3	131	2.24
*B*2_7	109	126	123	99	0.14	11.97	14	113	8.13
*B*3_3	163	176	168	159	0.1	4.47	8	167	0.6
*B*3_8	172	193	183	164	0.13	8.22	15	179	2.19
*B*5_6	96	111	101	87	0.09	11.03	11	98	2.97
*B*5_7	108	123	116	105	0.17	7.55	9	114	1.72
*B*5_8	110	121	118	116	0.17	0.8	1	117	0.85
*B*5_9	101	110	105	100	0.11	3.69	4	104	0.95
*B*6_2	140	159	–	95	0.05	–	–	–	–
*B*6_7	161	174	167	146	0.08	11.6	19	165	1.2
*B*6_8	201	243	–	192	0.22	–	–	–	–
*B*8_2	94	109	99	74	0.08	23.48	23	97	2.02
*B*8_3	128	179	170	114	0.31	23.16	30	144	15.29
*B*8_6	119	130	127	118	0.19	5.35	7	125	1.57
*B*9_6	167	181	173	163	0.08	4.96	9	172	0.58
*B*9_7	155	173	160	139	0.06	12.42	20	159	0.63
*B*11_2	116	133	130	114	0.34	9	11	125	3.85
*B*12_9	174	187	180	170	0.09	4.82	9	179	0.56
*B*12_10	188	221	213	143	0.15	27.94	53	196	7.98

Table 6: *The NS-algorithm applied to the set of 21 problems (with $\beta < 1.0$) not solved to optimality.*

generating all possible cutting patterns). By utilizing the NS-algorithm directly for the remaining 21 or 17.5% of the problems the gap between the computed lower and upper bounds on N is less than 4% for 16 out of the 21 problems. The experience gained with this set of data and the NS-/OC-algorithms suggest that practitioners may prefer to run complex problems until an acceptable gap is obtained.

This research demonstrates that this type of approach may help to solve problems that were considered intractable thus far. Giving attention to methods that employ the better management and role of β during the generation of cutting patterns may prove to be worthwhile to improve the efficiency. Less attention was given here to these aspects since the main goal was to investigate the feasibility of the NS-algorithm. It may also be worthwhile to extend these ideas to solution methods for the 2DMSSCSP.

8 Acknowledgements

The authors would like to thank the referees for their constructive comments and suggestions.

References

[1] Amaral ARS & Wright M, 2001, *Efficient algorithm for the constrained two-dimensional cutting stock problem*, International Transactions in Operational Research, **8(1)**, pp. 3–13.

[2] Beasley JE, 1985, *An algorithm for the two-dimensional assortment problem*, European Journal of Operational Research, **19(2)**, pp. 253–261.

[3] Beasley JE, 1985, *Algorithms for unconstrained two-dimensional guillotine cutting*, Journal of the Operational Research Society, **36(4)**, pp. 297–306.

[4] Christofides N & Whitlock C, 1977, *An algorithm for two dimensional cutting problems*, Operations Research, **25(1)**, pp. 30–44.

[5] Cintra CF, Miyazawa FK, Wakabayashi Y & Xavier EC, 2008, *Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation*, European Journal of Operational Research, **191(1)**, pp. 61–85.

[6] Gilmore PC & Gomory RE, 1961, *A linear programming approach to the cutting-stock problem*, Operations Research, **9(6)**, pp. 849–859.

[7] Gilmore PC & Gomory RE, 1963, *A linear programming approach to the cutting-stock problem Part II*, Operations Research, **11(6)**, pp. 863–888.

[8] Gilmore PC & Gomory RE, 1965, *Multistage cutting stock problems of two and more dimensions*, Operations Research, **13(1)**, pp. 94–120.

[9] Herz JC, 1972, *A recursive computational procedure for two-dimensional stock-cutting*, IBM Journal of Research and Development, **16(5)**, pp. 462–469.

[10] IBM ILOG CPLEX V12.1, *User's Manual for CPLEX*, 2009, Available at: ftp://public.dhe.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps_usrmancplex.pdf.

[11] Oliveira JE & Ferreira JS, 1990, *An improved version of Wang's algorithm for two-dimensional cutting problems*, European Journal of Operational Research, **44(2)**, pp. 256–266.

[12] Steyn T & Hattingh JM, 2011, *Notes and experiments with Amaral's algorithm for generating cutting patterns*, (Unpublished) Technical report: Research Unit for BMI, Potchefstroom Campus, North West University, FABWI-N-RKW: 2011-271. (2011-02-08).

[13] Vasko FJ, 1989, *A computational improvement to Wang's two-dimensional cutting stock algorithm*, Computers and Industrial Engineering, **16(1)**, pp. 109–115.

[14] Vasko FJ & Bartkowski CL, 2009, *Using Wang's two-dimensional cutting stock algorithm to solve difficult problems*, International Transactions in Operational Research, **16(6)**, pp. 829–838.

[15] Wang PY, 1983, *Two algorithms for constrained two dimensional cutting stock problems*, Operational Researchs, **31(3)**, pp. 573–586.

[16] Wäscher G, Haussner H & Schumann H, 2007, *An improved typology of cutting and packing problems*, European Journal of Operational Research, **183(3)**, pp. 1109–1130.